ABSTRACT

Title of dissertation: LOW-RANK SOLUTION METHODS FOR DISCRETE PARAMETRIZED PARTIAL DIFFERENTIAL EQUATIONS

Tengfei Su
Doctor of Philosophy, 2019

Dissertation directed by: Professor Howard C. Elman
Department of Computer Science

Stochastic partial differential equations are widely used to model physical problems with uncertainty. For numerical treatment, the stochastic Galerkin discretization in general gives rise to large, coupled algebraic systems that are computationally expensive to solve. In this thesis, we develop efficient iterative algorithms to reduce the costs, by taking advantage of the structures of the systems and computing low-rank approximations to the discrete solutions.

We demonstrate this idea by exploring three types of problems: (i) the stochastic diffusion equation, in which the diffusion coefficient is a random field; (ii) a collection of stochastic eigenvalue problems arising from models of diffusion and fluid dynamics; (iii) stochastic version of the time-dependent incompressible Navier–Stokes equations with an uncertain viscosity. These problems range from a relatively straightforward linear elliptic problem for which we are able to obtain rigorous results on convergence rates for solvers, to more complex models that include eigenvalue computations and nonlinear and time-dependent computations.

For the diffusion problem, we propose a low-rank multigrid method for solving the linear system obtained from the stochastic Galerkin discretization. In the algorithm, the iterates are represented as low-rank matrices, with which the associated computations become much cheaper. We conduct a rigorous error analysis for the convergence of the low-rank multigrid method. Numerical experiments show significant cost savings from low-rank approximation.

We design a low-rank variant of the inverse subspace iteration algorithm for stochastic eigenvalue problems. We apply low-rank iterative methods to efficiently solve the large algebraic systems required at each step of the algorithm, and show that the costs of other computations, including the Gram–Schmidt process and the Rayleigh quotient, are also greatly reduced. The accuracy of the solutions and efficiency of the algorithm are illustrated in numerical tests.

For the time-dependent Navier–Stokes problem, we consider an all-at-once formulation where the discrete solutions at all the time steps are represented in a three-dimensional tensor. In the nonlinear iteration, we compute low-rank tensor approximations to explore further reduction in memory and computation. Effective mean-based preconditioners are derived for the all-at-once systems. The low-rank algorithm is able to efficiently handle large-size problems.

LOW-RANK SOLUTION METHODS FOR DISCRETE PARAMETRIZED PARTIAL DIFFERENTIAL EQUATIONS

by

Tengfei Su

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2019

Advisory Committee:
Professor Howard C. Elman, Chair/Advisor
Professor James Baeder
Professor Lise-Marie Imbert-Gérard
Professor Ricardo H. Nochetto
Professor Konstantina Trivisa

Acknowledgments

It still feels like yesterday when I came to the states for graduate school in the fall of 2014 although almost five years have passed. I owe my sincere gratitude to all the people who have helped me during my PhD life and who made this thesis possible.

Fist of all, I am deeply in debt to my advisor, Howard Elman. It is because of his invaluable guidance, generous support, and constant patience that my graduate study has been smooth and fruitful. I appreciate the time and effort he has put in this work, from my choice of research topics to the detailed editing of the thesis. I learnt from him not only different aspects of the research work, but also many of his wisdoms in life. It is my greatest fortune to have him as my advisor.

I would like to thank my committee, James Baeder, Lise-Marie Imbert-Gérard, Ricardo Nochetto, and Konstantina Trivisa for their time and for their support of this work. Dr. Nochetto and Dr. Trivisa gave me a lot of help and advice at the early stage of my PhD, which made my transition to graduate school much easier.

Many thanks to my dearest friends at Maryland. I will miss the lovely conversations, delicious foods, and wonderful trips shared with them. I am grateful to have their company to share joys and difficulties. They made my life colorful here.

Most importantly, none of my academic achievements would have been possible without the endless love from my family. I owe much to my parents for their unconditional support for whatever choice I make for my life.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

AMG	Algebraic multigrid
BiCGstab	Biconjugate gradient stabilized method
CG	Conjugate gradient method
DMRG	Density matrix renormalization group method
GMG	Geometric multigrid
GMRES	Generalized minimal residual method
gPC	Generalized polynomial chaos
HT	Hierarchical Tucker
KL	Karhunen–Loève
LOBPCG	Locally optimal block preconditioned conjugate gradient method
LSC	Least-squares commutator
MC	Monte Carlo
MINRES	Minimum residual method
PCD	Pressure convection-diffusion
PDE	Partial differential equation
SG	Stochastic Galerkin
SVD	Singular value decomposition
TT	Tensor train

Chapter 1: Introduction

This thesis is devoted to developing efficient computational algorithms for solving physical problems modeled as stochastic partial differential equations (PDEs), based on low-rank approximation techniques for model order reduction. We start the introduction with a review of uncertainty quantification for stochastic PDEs and the computational approaches. Then we discuss the existing methods that explore low-rank structures in the stochastic problems for handling large problem sizes or complicated physical models. In the end we summarize the main contributions of this thesis and provide an outline for the following chapters.

1.1 Background

Many physical problems are modeled as stochastic PDEs, where some random inputs are used to characterize possible variations in the physical properties, boundary conditions, or source terms, due to measurement errors, imprecise knowledge about the systems, or intrinsic variability. For example, in models of diffusion, a stochastic diffusion coefficient can be used to describe the permeability of a heterogeneous porous medium, or, in models of fluid dynamics, uncertain boundary conditions depending on some random variables can be used to study the effect

of fluctuations in the temperature of walls in a Boussinesq cavity model [59]. Let $(\Omega, \mathcal{F}, P)$ be a probability space and $\omega \in \Omega$. In general, a stochastic model can be abstractly written as

$$\mathcal{L}(a(\omega); u(\omega)) = 0, \tag{1.1}$$

where $\mathcal{L}$ represents a PDE with appropriate forcing terms and boundary conditions, and the uncertainty is incorporated in the input data $a(\omega)$. As a result, the solution $u(\omega)$ of the equation is also a stochastic function. In a forward propagation problem, the probability distribution about the input $a(\omega)$ is assumed known, and the objective of uncertainty quantification is to determine the distribution of the solution, or provide reliable predictions about probabilities of certain events associated with the output of the model. For an inverse problem, some observations about the output are given and the goal is to identify the corresponding input parameters or estimate certain state variables [91]. In this thesis, we will focus on the forward problem and develop efficient algorithms for computing the solution $u(\omega)$.

A straightforward approach for handling the forward problem is the sampling-based Monte Carlo (MC) method. Random samples are drawn from the given distribution of the input. For each realization of the input data $a(\omega^{(r)})$, a deterministic equation is solved for the corresponding solution $u(\omega^{(r)})$. The mean and variance of the solution can be estimated via

$$\mathbb{E}[u] \approx \frac{1}{n_r} \sum_{r=1}^{n_r} u(w^{(r)}), \quad \mathrm{Var}[u] \approx \frac{1}{n_r - 1} \sum_{r=1}^{n_r} (u(w^{(r)}) - \mathbb{E}[u])^2, \tag{1.2}$$

where n_r is the sample size. Probabilities of events can also be easily computed from the sample solutions. However, it is known that the Monte Carlo method suffers

from slow convergence. For instance, the error for the mean estimate behaves as $O(n_r^{-1/2})$ (see, e.g. [62]). Therefore, a large number of samples are required to obtain accurate results, and the associated computational cost can be prohibitive especially when the problem is already nontrivial to solve in a deterministic setting. To address this high cost, quasi-Monte Carlo methods [17] and multilevel Monte Carlo methods [35] have been studied for improved convergence.

In this thesis, we focus on a different type of approach, the stochastic Galerkin (SG) method. Instead of solving a deterministic PDE for each sample, the stochastic Galerkin approach computes a surrogate as an approximation to the true solution function. The surrogate is typically expressed as a series

$$u(\omega) = \sum_{s=1}^{\infty} u_s \psi_s(\omega), \tag{1.3}$$

where $\{\psi_s(\omega)\}$ is an appropriately selected set of basis functions, and $\{u_s\}$ are deterministic coefficients. In practice, a finite number of terms are used, and the coefficients are determined by imposing an orthogonality condition. Once the surrogate is available, it becomes a much easier task to compute the statistics and other properties of the solution, either analytically or by sampling the basis functions. The method also enjoys fast convergence rates under some mild smoothness conditions on the random data [3, 15]. For example, it was shown in [3] that when the input data is parametrized with a finite number of random variables, the error associated with the surrogate decreases exponentially with respect to the polynomial degree of the basis functions. The stochastic Galerkin approach has received increasing attention since it was first studied in [34] for applications in structural mechanics.

It has been applied to numerous problems in fluid dynamics, ranging from flows in porous media to complex thermofluid and reacting flows [53, 68, 97].

To compute the surrogate, the stochastic Galerkin method typically gives rise to a large, coupled algebraic system, whose size might be orders of magnitude larger than those obtained from the deterministic subproblems in the Monte Carlo method. Solving such systems can be computationally challenging and requires new versions of iterative solution algorithms, such as preconditioned Krylov subspace methods. Efficient solution methods have been studied by exploiting the Kronecker product structures or blockwise sparsity patterns of the systems, so the computational effort required becomes much less that the problem size suggests [21, 73]. For a linear stochastic diffusion problem, a geometric multigrid method was proposed in [23], and a mean-based preconditioner was studied in [74]. Both were shown to give numbers of iterations independent of some of the discretization parameters. In an acoustic scattering model with uncertainty in the forcing terms or boundary conditions, the authors in [25] reduced the algebraic system to a much smaller size with multiple right-hand sides, and applied block Krylov methods to efficiently solve the system. Nonlinear iterative methods and effective preconditioners have been studied in [75, 87] for the Navier–Stokes equations with random input data. The goal of this thesis is to address the computational difficulty for solving the algebraic systems associated with the stochastic Galerkin method for stochastic PDEs, by using the idea of low-rank approximation for the discrete solutions.

1.2 Existing work

In this section we briefly survey some recent developments of iterative methods with low-rank approximation techniques for solving stochastic PDE problems. As discussed earlier, the stochastic Galerkin discretization results in large algebraic systems that are computationally expensive to solve. Such large systems also arise from discretization of PDEs with high spatial or stochastic dimensions. Low-rank approximation techniques are used to reduce the high storage and computational costs for solving large-scale problems, and the basic idea is to employ a compressed data representation in classical iterative algorithms, and repeatedly apply low-rank truncation for data compression [38]. Let F define an iterative method

$$\mathbf{u}_{k+1} = F_k(\mathbf{u}_k), \tag{1.4}$$

where $\mathbf{u}_k$ is a vector representation of the approximate solution. For example, $\mathbf{u}_k$ may be the coefficients of the basis functions of a stochastic Galerkin discretization. In general, for the discrete solution of a stochastic PDE, the vector can be equivalently cast as a multidimensional array, or tensor (see specific examples in chapters 3 to 5). Let $\mathcal{U}_k$ be a low-rank approximation to the tensor, then the iterative algorithm becomes

$$\mathcal{U}_{k+1} = \mathcal{T}(F_k(\mathcal{U}_k)), \tag{1.5}$$

where $\mathcal{T}$ is a low-rank truncation operator. A two-dimensional tensor reduces to a matrix and the low-rank approximation can be directly computed from a singular value decomposition (see section 2.5). In higher dimensions, many different low-rank

formats have been developed. Examples include the CANDECOMP/PARAFAC (CP) decomposition [55], the tensor train (TT) decomposition [70], and the hierarchical Tucker (HT) decomposition [37]. These low-rank tensor formats use much less storage than the full tensor, and the basic operations associated with the tensors also become much cheaper. In the course of the iteration, the tensor ranks may grow very quickly, from computations such as summations and pointwise products. The truncation operator $\mathcal{T}$ is thus needed to reduce the tensor ranks to maintain efficiency. There have been many studies in designing low-rank iterative methods with applications to various physical models [2, 4, 5, 7–10, 18, 41, 56, 57, 60, 65]. Some theoretical results are also available on the existence of low-rank solutions and convergence of the low-rank methods [7, 42, 51, 56, 65].

For stochastic linear elliptic PDEs, a preconditioned Richardson method with low-rank matrix approximation was studied in [65]. Convergence results are available for such cases where the low-rank iterative method can be expressed as a perturbed fixed point iteration [42, 65]. However, for more complicated iterative algorithms, such as Krylov subspace methods, the effect of truncation on the convergence is much harder to analyze. The preconditioned conjugate gradient and BiCGstab methods have been studied in [56] to compute low-rank matrix and HT approximations to the solutions of linear elliptic PDEs with parametrized or stochastic coefficients. The iterates are compressed with a relative tolerance on the singular values or a maximal rank constraint, and the accuracy attained by the low-rank methods is closely related to these choices. In [4] the authors proposed a low-rank GMRES-like method for solving linear systems from discretization of PDEs in high

spatial dimensions. The residual of the system is projected onto a subspace spanned by vectors in low-rank HT format, and the iterates are truncated to a fixed rank to reduce computational complexity. The method is also applicable for discrete stochastic PDEs.

A low-rank solver for time-dependent diffusion equations with stochastic coefficients was developed in [7]. With an implicit Euler method for time discretization, a linear system is solved at each time step using a low-rank conjugate gradient method. The authors showed in theory that the discrete solution can be well approximated by a low-rank matrix under certain conditions. People have also explored low-rank reduction in time with an "all-at-once" formulation, where the equations at all the discrete time steps are collected to form a single system. Such a formulation arises naturally from optimal control problems with time-dependent PDE constraints [8, 90], where the solution is required over the whole time horizon. The storage increases dramatically with the number of time steps, and a low-rank approximation becomes essential for efficient computations. In [8] the authors used a preconditioned MINRES method combined with low-rank TT format for solving the saddle-point systems obtained from the discrete optimality condition of optimal control problems constrained by unsteady PDEs with random inputs.

Low-rank iterative methods have been developed to solve PDE-related eigenvalue problems. Similar to low-rank Krylov subspace methods for solving linear systems, low-rank approximation techniques can be combined with an iterative eigensolver. A low-rank variant of the LOBPCG method was proposed in [57] to compute the smallest eigenvalue of a symmetric, high-dimensional discrete PDE operator.

The HT format was used to overcome the exponential growth of degrees of freedom in high spatial dimensions. A low-rank Arnoldi method was discussed in [10] to compute compressed representation of the dominant eigenvectors of posterior covariance matrices in the context of Bayesian inverse problems. In [9], the authors formulated the stochastic eigenvalue problem as a nonlinear algebraic system with stochastic Galerkin discretization, and used an inexact Newton method to compute the surrogate eigenvalue closest to the initial guess. A low-rank BiCGstab method was applied to efficiently solve the linear system associated with the Jacobian matrix at each nonlinear step.

1.3 Contributions of this thesis

In this thesis, we develop efficient low-rank iterative algorithms to solve the algebraic systems obtained from stochastic Galerkin discretization of stochastic PDEs. We consider three types of problems: a stochastic diffusion equation with random diffusion coefficients, stochastic eigenvalue problems arising from models of diffusion and fluid dynamics, and time-dependent incompressible Navier–Stokes equations with uncertain inputs. The main contributions are summarized as follows.

First, for the stochastic diffusion problem, we propose a low-rank multigrid method. The stochastic Galerkin discretization of the equation gives rise to a large linear system with Kronecker product structure. Numerically we show that the discrete solution, when represented as a matrix, exhibits exponential decay in its singular values, and thus can be well approximated by a low-rank matrix. In the

proposed algorithm, the iterates are represented as low-rank matrices to reduce the memory and computational costs. Theoretically, we conduct a rigorous convergence analysis for the low-rank multigrid method. As we have discussed, such results are not available for general Krylov subspace methods. It shows the error introduced by the low-rank approximation, and provides insights on how to choose the truncation strategies in the algorithm. Numerical experiments show significantly improved efficiency of the low-rank solver compared against the original multigrid method (without low-rank approximation), especially when the number of degrees of freedom associated with the spatial discretization is large. This is discussed in chapter 3.

For stochastic eigenvalue problems, we design a low-rank variant of the inverse subspace iteration algorithm for computing one or several minimal eigenvalues and corresponding eigenvectors. The eigenvalue problems arise from discretization of self-joint PDEs with random data. The algorithm computes low-rank approximations to the eigenvectors. A large linear system is solved at each step of the algorithm, with a low-rank multigrid or Krylov subspace method. Other computations required for subspace iteration including a Gram–Schmidt process and construction of a Rayleigh quotient are also inexpensive in the low-rank format. We test the proposed algorithm on two specific problems, a stochastic diffusion problem with some poorly separated eigenvalues, and an operator derived from a discrete stochastic Stokes problem whose minimal eigenvalue is related to the inf-sup stability constant. For the diffusion problem we show that a Rayleigh–Ritz procedure can be used to improve the accuracy of the solution. Numerical experiments show that the low-rank method produces accurate solutions with reduced computational

costs. See chapter 4 for detailed discussions.

The Navier–Stokes equations is a challenging problem where one has to handle nonlinearity, time-dependency, as well as stochasticity. We consider an all-at-once formulation where the equations at all the discrete time steps are collected in a single system. Low-rank tensor approximations are used to overcome the high memory requirements. At each nonlinear step, the all-at-once system is solved by a low-rank GMRES method. We derive mean-based preconditioners for the system, using state-of-the-art results for deterministic problems. With an inexact nonlinear iteration and effective preconditioners, the GMRES method is efficient in solving the large systems and only requires a small number of iterations to reach the required accuracy. The matrix-vector product computation introduces large rank increases for the intermediate iterates, and we show that an approximate convection matrix can be used to get rid of this problem. The proposed low-rank algorithm allows us handle large problem sizes. Numerical experiments show the effectiveness of the preconditioners and the memory and computational savings from the low-rank tensor approximation. This will be discussed in chapter 5.

Chapter 2: Preliminaries

In this chapter we review the basic procedure of the stochastic Galerkin method, and summarize some of the classical iterative algorithms that will be useful in later discussions. Some facts about low-rank approximation are also given.

2.1 Stochastic Galerkin method

2.1.1 Input parametrization

In this thesis we will focus on stochastic PDEs with uncertainty $a(\omega)$ coming from some physical properties associated with the models represented by the PDEs. We start with a discussion on the representation of the random input $a(\omega)$. Depending on the problems studied, a simple choice can be just a single random variable. However, in many cases, the random inputs are stochastic fields $a(x, \omega)$, where x is the spatial parameter. For practical computation, a finite dimensional approximation is often required. This can be achieved using a truncated Karhunen–Loève (KL) expansion [34, 61]. Let $(\Omega, \mathcal{F}, P)$ be a probability space, and let $\mathcal{D}$ be a spatial domain. For a second order stochastic process $a(x, \omega) : \mathcal{D} \times \Omega \to \mathbb{R}$ where

$\mathbb{E}[a^2] < \infty$ for all $x \in \mathcal{D}$, the KL expansion has the form

$$a(x,\omega) = a_0(x) + \sum_{l=1}^{\infty} \sqrt{\beta_l} a_l(x)\xi_l(\omega), \tag{2.1}$$

where $a_0(x)$ is the mean function, and $(\beta_l, a_l(x))$ is the lth eigenpair of the covariance function $c(x,y) = \mathbb{E}[(a(x,\omega) - a_0(x))(a(y,\omega) - a_0(y))]$ such that

$$\int_{\mathcal{D}} c(x,y)a_l(y)\mathrm{d}y = \beta_l a_l(x). \tag{2.2}$$

The eigenfunctions $\{a_l(x)\}$ are orthonormal in $L^2(\mathcal{D})$, and the eigenvalues $\{\beta_l\}$ are in nonincreasing order. The random variables $\{\xi_l(\omega)\}$ are mutually uncorrelated, satisfying

$$\mathbb{E}[\xi_l] = 0, \quad \mathbb{E}[\xi_i \xi_j] = \delta_{ij}, \tag{2.3}$$

and they are defined via

$$\xi_l(\omega) = \frac{1}{\sqrt{\beta_l}} \int_{\mathcal{D}} (a(x,\omega) - a_0(x))a_l(x)\mathrm{d}x. \tag{2.4}$$

To simplify the computations, we will also assume them to be independent and identically distributed.

The KL expansion in eq. (2.1) is an infinite series and for approximation we use a truncated version with a finite number of random variables,

$$a(x,\omega) \approx a_0(x) + \sum_{l=1}^{m} \sqrt{\beta_l} a_l(x)\xi_l(\omega). \tag{2.5}$$

The number of terms m required depends on the decay of the eigenvalues $\{\beta_l\}$. For example, for the exponential covariance $c(x,y) = \exp(-\|x-y\|_1/b)$, if the correlation length b is larger, the eigenvalues decay faster (see fig. 2.1), and a smaller number of terms are needed in eq. (2.5) to obtain a good approximation. Now with eq. (2.5),

the input is parametrized with a finite-dimensional random vector $\xi : \Omega \to \Gamma \subset \mathbb{R}^m$ and can be written as $a(x, \xi)$.

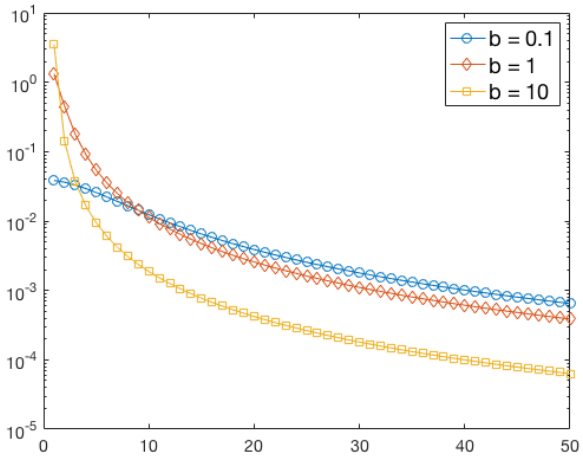


Figure 2.1: First 50 eigenvalues of the exponential covariance on a spatial domain $\mathcal{D} = [-1, 1]^2$ with different correlation lengths b.

2.1.2 Generalized polynomial chaos

The generalized polynomial chaos (gPC) [95, 96] provides an expansion for any second order stochastic process $u(x, \omega) : \mathcal{D} \times \Omega \to \mathbb{R}$ and a way to construct finite-term approximations, with orthogonal multi-dimensional polynomials of independent random variables. As the number of the random variables and the order of the polynomials go to infinity, the expansion converges to the true process in the mean square sense [96]. When the input in eq. (1.1) is parametrized with a random vector $\xi : \Omega \to \Gamma \subset \mathbb{R}^m$, the solution is also dependent on ξ and can be written as $u(x, \xi) : \mathcal{D} \times \Gamma \to \mathbb{R}$. In such cases, the number of the random variables is fixed as

m. The gPC expansion of the solution is in the following form

$$u(x,\xi) = \sum_{s=1}^{\infty} u_s(x)\psi_s(\xi). \tag{2.6}$$

Let $(\Gamma, \mathcal{B}, \mu)$ be the probability space induced by ξ, where μ is the induced measure. The functions $\{\psi_s(\xi)\}$ form an orthogonal basis for $L^2(\Gamma, \mu)$, and after normalization they satisfy

$$\langle \psi_r(\xi), \psi_s(\xi) \rangle = \int_{\Gamma} \psi_r(\xi)\psi_s(\xi)\mathrm{d}\mu = \delta_{rs}. \tag{2.7}$$

The specific forms of the gPC basis functions depend on the distribution of the random variables ξ. For example, if each of the random variables has a standard normal distribution, the basis functions are multi-dimensional Hermite polynomials. See table 2.1 for some commonly used distributions and corresponding gPC polynomials. To construct a finite-term approximation to the stochastic process, one can restrict the total degree of the polynomials (i.e., the sum of the degrees of univariate polynomials) to p. Then

$$u(x,\xi) = \sum_{s=1}^{n_\xi} u_s(x)\psi_s(\xi) \tag{2.8}$$

and the number of terms $n_\xi = (m+p)!/(m!p!)$. For example, Let $m = 2$, $p = 3$. Let the multi-index $\alpha = (\alpha_1, \alpha_2)$ denote the degrees of polynomials of the two random variables ξ_1 and ξ_2. Then the possible values of α are $(0,0)$, $(1,0)$, $(0,1)$, $(2,0)$, $(1,1)$, $(0,2)$, $(3,0)$, $(2,1)$, $(1,2)$, and $(0,3)$. In the case of Gaussian random variables, the (unnormalized) univariate Hermite polynomials are $H_0(t) = 1$, $H_1(t) = t$, $H_2(t) = t^2 - 1$, and $H_3(t) = t^3 - 3t$, and the basis functions $\{\psi_s(\xi)\}$ in eq. (2.8) are

$$1, \xi_1, \xi_2, \xi_1^2 - 1, \xi_1\xi_2, \xi_2^2 - 1, \xi_1^3 - 3\xi_1, (\xi_1^2 - 1)\xi_2, \xi_1(\xi_2^2 - 1), \xi_2^3 - 3\xi_2. \tag{2.9}$$

In general, a small value of p suffices for a good approximation of the stochastic process due to the fast convergence of the sequence in eq. (2.8) [96].

Table 2.1: Univariate random variable distributions and corresponding orthogonal polynomials.

Distribution	Polynomials	Support
Gaussian	Hermite	$(-\infty, \infty)$
gamma	Laguerre	$[0, \infty)$
beta	Jacobi	$[t_1, t_2]$
uniform	Legendre	$[t_1, t_2]$

The stochastic Galerkin method [34] for eq. (1.1) computes a surrogate solution, expanded with gPC basis functions as in eq. (2.8). Define the residual associated with the surrogate as

$$\mathcal{R} = \mathcal{L}\big(a(x,\xi); \sum_{s=1}^{n_\xi} u_s(x)\psi_s(\xi)\big). \tag{2.10}$$

This is in general not equal to zero. A Galerkin projection is applied so that the residual is orthogonal to the space spanned by the basis functions $\{\psi_s(\xi)\}$, i.e.,

$$\Big\langle \mathcal{L}\big(a(x,\xi); \sum_{s=1}^{n_\xi} u_s(x)\psi_s(\xi)\big), \psi_r(\xi) \Big\rangle = 0, \quad r = 1, \ldots, n_\xi. \tag{2.11}$$

This results in n_ξ coupled equations for the expansion coefficients $\{u_s(x)\}$. Equation (2.11) can be combined with a finite element method for the spatial discretization. Time discretization can also be carried out for unsteady problems. In the following chapters we will consider such formulations for some specific stochastic PDEs. More details will be given as we discuss the stochastic diffusion problem in chapter 3.

2.2 Iterative solvers for linear systems

The stochastic Galerkin discretization in general gives rise to a large algebraic system for the coefficients of the surrogate solution. Linear system solves are required from discretization of linear stochastic PDEs or inside a nonlinear iterative algorithm. For large-size linear systems, a direct method based on Gaussian elimination or matrix decompositions becomes too expensive to be feasible. In such situations, iterative methods are used. The major cost in an iterative solver, the matrix-vector product computation, is order $O(n)$ for a sparse matrix, where n is the problem size. Fast convergence can be achieved with good preconditioners. In this chapter, we briefly review Krylov subspace methods and multigrid methods.

2.2.1 Krylov subspace methods

Consider a linear system $A\mathbf{u} = \mathbf{f}$, where $A \in \mathbb{R}^{n\times n}$ is nonsingular. Krylov subspace methods seek an approximate solution $\mathbf{u}_k$ in the space $\mathbf{u}_0 + \mathcal{K}_k(A, \mathbf{r}_0)$, where $\mathcal{K}_k$ is a k-dimensional Krylov subspace defined as

$$\mathcal{K}_k(A, \mathbf{r}_0) = \text{span}\{\mathbf{r}_0, A\mathbf{r}_0, A^2\mathbf{r}_0 \ldots, A^{k-1}\mathbf{r}_0\}, \tag{2.12}$$

$\mathbf{u}_0$ is an initial guess for the solution, and $\mathbf{r}_0 = \mathbf{f} - A\mathbf{u}_0$ is the corresponding residual. In other words, the approximate solution is in the form

$$\mathbf{u}_k = \mathbf{u}_0 + q_{k-1}(A)\mathbf{r}_0 \tag{2.13}$$

where q_{k-1} is a polynomial of degree $k-1$. The accuracy of the approximate solution $\mathbf{u}_k$ is improved by successively increasing the Krylov subspace dimension k.

An optimality condition can be imposed by requiring the residual $\mathbf{r}_k = \mathbf{f} - A\mathbf{u}_k$ to be orthogonal to some space $\mathcal{M}_k$. The following two choices of $\mathcal{M}_k$ result in some of the most successful iterative solvers [26, 78]:

1. $\mathcal{M}_k = \mathcal{K}_k$. If A is symmetric positive definite, this is equivalent to minimizing $\|\mathbf{e}_k\|_A = (A(\mathbf{u} - \mathbf{u}_k), \mathbf{u} - \mathbf{u}_k)^{1/2}$, and it gives the conjugate gradient (CG) method [45].

2. $\mathcal{M}_k = A\mathcal{K}_k$. This is equivalent to minimizing $\|\mathbf{r}_k\|_2$, and the resulting algorithms are the minimum residual (MINRES) method [72] for a symmetric matrix A and the generalized minimal residual (GMRES) method [80] for a general nonsingular A.

The convergence of Krylov subspace methods depends on properties of the coefficient matrix A, such as its spectrum. Preconditioning techniques are used to improve the eigenvalue distribution or to reduce the condition number of the system so that the iterative algorithms have faster convergence, and thus lower computational cost. For instance, a right-preconditioned system has the form

$$AP^{-1}\mathbf{v} = \mathbf{f}, \quad \mathbf{u} = P^{-1}\mathbf{v}, \tag{2.14}$$

where the preconditioner P should be a good approximation to A, and it should be inexpensive to apply the action of P^{-1} for a given vector. Krylov subspace methods are now used to solve the system associated with AP^{-1} instead of the original A. We will discuss constructions of preconditioners for specific problems in the following chapters.

2.2.2 Multigrid methods

Multigrid methods [14, 40] can be divided to geometric multigrid (GMG) and algebraic multigrid (AMG). In this section we focus on GMG where the linear system is solved on a hierarchy of spatial grids. AMG, on the other hand, only uses information from the entries of the coefficient matrix but ignores the physical discretization. We refer to [13, 77] for more details about AMG.

The two ingredients of GMG are the smoothing operator and the coarse-grid correction. The smoothing operator is a stationary iterative method, based on a splitting of the coefficient matrix $A = M - N$, such that

$$M\mathbf{u}_{k+1} = N\mathbf{u}_k + \mathbf{f}, \quad \text{or} \quad \mathbf{u}_{k+1} = \mathbf{u}_k + M^{-1}(\mathbf{f} - A\mathbf{u}_k). \tag{2.15}$$

Examples are the Jacobi method and the Gauss–Seidel method. Such methods are effective in eliminating the oscillatory components of the error, but the smooth components get damped slowly. After a few steps of the stationary iteration, the error becomes smooth and can be well represented on a coarse grid. The system is then transferred to the coarse grid and solved much less expensively. Specifically, let superscripts h and $2h$ denote quantities associated with the fine grid and the coarse grid. The prolongation operator I_{2h}^h and the restriction operator I_h^{2h} define how a function (represented as a vector via finite element discretization) is mapped from the coarse grid to the fine grid, and vice versa. A two-grid scheme $\mathbf{v}^h \leftarrow \mathrm{MG}(\mathbf{v}^h, \mathbf{f}^h)$ is given as follows.

- Smoothing: for ν_1 steps $\mathbf{v}^h \leftarrow \mathbf{v}^h + M^{-1}(\mathbf{f}^h - A^h\mathbf{v}^h)$.

- Residual restriction: $\mathbf{r}^{2h} = I_h^{2h}(\mathbf{f}^h - A^h\mathbf{v}^h)$.
- Coarse-grid correction: solve $A^{2h}\mathbf{e}^{2h} = \mathbf{r}^{2h}$.
- Prolongation and update: $\mathbf{v}^h \leftarrow \mathbf{v}^h + I_{2h}^h\mathbf{e}^{2h}$.
- Smoothing: for ν_2 steps $\mathbf{v}^h \leftarrow \mathbf{v}^h + M^{-1}(\mathbf{f}^h - A^h\mathbf{v}^h)$.

On the coarse grid, the coefficient matrix A^{2h} can be built from $A^{2h} = I_h^{2h}A^hI_{2h}^h$. In some cases this is equivalent to assembling A^{2h} directly from the spatial discretization on the coarse grid. The above procedure can be applied recursively to obtain the multigrid version. That is, for the coarse-grid correction, the system is solved with the same scheme $\mathbf{e}^{2h} \leftarrow \mathrm{MG}(\mathbf{e}^{2h}, \mathbf{r}^{2h})$.

2.3 Numerical methods for eigenvalue problems

Another important problem class we consider are eigenvalue problems. In this section we review two iterative algorithms, the inverse subspace iteration method and a variant of the Lanczos algorithm known as the locally optimal preconditioned conjugate gradient method, for solving a deterministic eigenvalue problem

$$A\mathbf{u}^s = \lambda^s\mathbf{u}^s, \quad s = 1, \ldots, n_e, \tag{2.16}$$

where $A \in \mathbb{R}^{n\times n}$ is symmetric, λ^s is the sth smallest eigenvalue and $\mathbf{u}^s$ is the corresponding eigenvector. We refer to [79, 89] for a thorough discussion. In chapter 4 we will develop variants of these algorithms for eigenvalue problems obtained from discretization of stochastic PDEs.

2.3.1 Inverse subspace iteration

When a single smallest eigenvalue is sought, the problem can be solved by the inverse iteration method, where the iterate $\mathbf{u}_k$ is given by

$$\mathbf{v}_k = A^{-1}\mathbf{u}_{k-1}, \quad \mathbf{u}_k = \frac{\mathbf{v}_k}{\|\mathbf{v}_k\|_2}. \tag{2.17}$$

For a large sparse matrix A, the action of A^{-1} can be achieved by applying the iterative solvers discussed in the previous section. The error associated with the approximate eigenvector $\mathbf{u}_k$ decreases as $O(\rho^k)$, where $\rho = |\lambda^1|/|\lambda^2|$. It indicates that the convergence is fast if λ^2 has a much larger modulus than λ^1. The inverse subspace iteration algorithm is a block generalization for computing more than one eigenvalue. For $U_k = [\mathbf{u}_k^1, \ldots, \mathbf{u}_k^{n_e}]$, the algorithm has the following steps.

- Compute $V_k = A^{-1}U_{k-1}$.
- Compute U_k as a Gram–Schmidt orthonormalization of V_k.

The second step can also be achieved by computing a QR factorization $V_k = Q_k R_k$ and setting $U_k = Q_k$. For the subspace iteration, the largest canonical angle between the approximate eigenspace spanned by the columns of U_k, and the true eigenspace, decreases essentially as $O(\rho^k)$, with $\rho = |\lambda^{n_e}|/|\lambda^{n_e+1}|$ (see, e.g., [89]).

2.3.2 LOBPCG method

The locally optimal block preconditioned conjugate gradient (LOBPCG) method [54] is an iterative method for solving a generalized eigenvalue problem $A\mathbf{u} =$

$\lambda B\mathbf{u}$, where both $A, B \in \mathbb{R}^{n\times n}$ are symmetric positive definite. Unlike the inverse subspace iteration algorithm, it does not require solving linear systems with A, but only matrix-vector products. The single eigenvalue version is based on successively minimizing the generalized Rayleigh quotient

$$\mathbf{u}_{k+1} = \min_{\mathbf{v}\in \mathrm{span}\{\mathbf{w}_k, \mathbf{u}_k, \mathbf{u}_{k-1}\}} \frac{\mathbf{v}^T A\mathbf{v}}{\mathbf{v}^T B\mathbf{v}}, \tag{2.18}$$

where $\mathbf{u}_k$ and $\mathbf{u}_{k-1}$ are the current and previous iterates, and $\mathbf{w}_k$ is the preconditioned residual. Let $\mu(\mathbf{v}) = \mathbf{v}^T A\mathbf{v}/\mathbf{v}^T B\mathbf{v}$. Then

$$\mathbf{w}_k = P^{-1}(A\mathbf{u}_k - \mu(\mathbf{u}_k)B\mathbf{u}_k), \tag{2.19}$$

where P is a left preconditioner for the eigenvalue problem. The preconditioner should be a good approximation to A, and the matrix $P^{-1}A$ has a much smaller condition number. As the algorithm converges, the vectors $\mathbf{u}_k$ and $\mathbf{u}_{k-1}$ become nearly linearly dependent. To increase stability, one can use an equivalent space, $\mathrm{span}\{\mathbf{w}_k, \mathbf{u}_k, \mathbf{p}_k\}$, where $\mathbf{p}_k$ is an implicitly computed difference between $\mathbf{u}_k$ and $\mathbf{u}_{k-1}$. The LOBPCG method is a block generalization of the above procedure. It is used to compute multiple eigenvalues simultaneously, or to accelerate the single eigenvalue computation. In general, the error associated with the computed eigenvalue λ_k decreases as $O(\rho^k)$, where ρ is an average convergence factor and it is related to the condition number of $P^{-1}A$.

2.4 Nonlinear iterative methods

In this section we summarize Picard's and Newton's methods for solving a nonlinear equation

$$F(\mathbf{u}) = \mathbf{0} \tag{2.20}$$

where $F : \mathbb{R}^n \to \mathbb{R}^n$ [50]. Picard's method is a fixed-point iteration. If the nonlinear equation can be expressed in the form $\mathbf{u} = K(\mathbf{u})$ for some $K : \mathbb{R}^n \to \mathbb{R}^n$, then the Picard iteration is given by

$$\mathbf{u}_{k+1} = K(\mathbf{u}_k). \tag{2.21}$$

It is known that if K is Lipschitz continuous with Lipschitz constant $\gamma < 1$, the fixed-point iteration converges linearly to the solution. We will consider this approach in the context of the incompressible Navier–Stokes equations, where the discrete problem has the form

$$\mathbf{u}_{k+1} = A^{-1}(\mathbf{u}_k)\mathbf{f}, \tag{2.22}$$

where $A(\mathbf{u}_k) \in \mathbb{R}^{n\times n}$ is an approximation to a Jacobian at $\mathbf{u}_k$. Each step of the Picard iteration requires solving a linear system. One way to save the computational work is to solve the linear systems inexactly. It is shown in [11] that Picard's method is guaranteed to converge if the following stopping criterion for eq. (2.22) is satisfied with $\tau < 1$,

$$\|A(\mathbf{u}_k)\mathbf{u}_{k+1} - \mathbf{f}\|_2 \leq \tau \|A(\mathbf{u}_k)\mathbf{u}_k - \mathbf{f}\|_2. \tag{2.23}$$

Although not used in the thesis, Newton's method is also included here for

completeness. For eq. (2.20), Newton's method is defined by

$$\mathbf{u}_{k+1} = \mathbf{u}_k + \mathbf{s}_k, \quad F'(\mathbf{u}_k)\mathbf{s}_k = -F(\mathbf{u}_k), \tag{2.24}$$

where $F'(\mathbf{u}_k) \in \mathbb{R}^{n\times n}$ is the Jacobian matrix at $\mathbf{u}_k$. Under some appropriate assumptions, Newton's method achieves quadratic convergence rate if the initial guess is close to the solution. In the cases where solving the linear systems at each Newton step is expensive, inexact Newton methods [16, 22] can be employed so that at each step $\mathbf{s}_k$ is solved to satisfy

$$\|F'(\mathbf{u}_k)\mathbf{s}_k + F(\mathbf{u}_k)\|_2 \leq \eta_k \|F(\mathbf{u}_k)\|_2. \tag{2.25}$$

If the forcing term $\eta_k < \eta_{\max} < 1$, the convergence is linear, and if η_k is chosen to be $O(\|F(\mathbf{u}_k)\|_2)$ then the convergence is still quadratic.

2.5 Low-rank approximation

Low-rank approximation finds applications in many different fields. In this thesis we will study low-rank approximations to the discrete solutions of stochastic PDEs in order to reduce the cost of iterative algorithms. For a two-dimensional array, or matrix $X \in \mathbb{R}^{n_1\times n_2}$, $n_1 \geq n_2$, the best rank κ approximation $\tilde{X}^*$ satisfies the following minimization problem,

$$\min \|X - \tilde{X}\|_F, \text{ s.t. } \operatorname{rank}(\tilde{X}) \leq \kappa. \tag{2.26}$$

The analytic solution is given by the Eckart–Young–Mirsky theorem [20] based on the singular value decomposition (SVD) of X

$$X = U\Sigma V^T = \begin{pmatrix} U_1 & U_2 \end{pmatrix} \begin{pmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{pmatrix} \begin{pmatrix} V_1^T \\ V_2^T \end{pmatrix} \tag{2.27}$$

where $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_{n_2})$ with the singular values in nonincreasing order, and $\Sigma_1 = \mathrm{diag}(\sigma_1, \ldots, \sigma_\kappa)$. Then the best approximation is

$$\tilde{X}^* = U_1 \Sigma_1 V_1^T \tag{2.28}$$

and

$$\|X - \tilde{X}^*\|_F = \sqrt{\sigma_{\kappa+1}^2 + \cdots + \sigma_{n_2}^2}. \tag{2.29}$$

For a multi-dimensional array, or tensor $\underline{z} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, $d \geq 3$, a low-rank approximation $\tilde{\underline{z}}$ can be constructed in different formats [38]. In chapter 5 we will use the tensor train (TT) decomposition [70]. The TT format for $\tilde{\underline{z}}$ is written as

$$\tilde{\underline{z}}(i_1, \ldots, i_d) = \sum_{\alpha_0, \ldots, \alpha_d} \tilde{\underline{z}}^{(1)}(\alpha_0, i_1, \alpha_1) \tilde{\underline{z}}^{(2)}(\alpha_1, i_2, \alpha_2) \cdots \tilde{\underline{z}}^{(d)}(\alpha_{d-1}, i_d, \alpha_d), \tag{2.30}$$

where the TT core $\tilde{\underline{z}}^{(j)}$ has size $\kappa_{j-1} \times n_j \times \kappa_j$, $\{\kappa_j\}_{j=1}^{d-1}$ are called TT ranks, and on the "boundary" $\kappa_0 = \kappa_d = 1$. Such a low-rank format can be computed from a so-called TT-SVD algorithm. It entails a sequence of SVDs to construct the best rank κ_j approximation (as defined in eq. (2.26)) of the unfolding matrix Z_j, where

$$Z_j(i_1, \ldots, i_j; i_{j+1}, \ldots, i_d) = \underline{z}(i_i, \ldots, i_d), \quad j = 1, \ldots, d-1; \tag{2.31}$$

i.e., the first j indices enumerate the rows of Z_j and the last $d-j$ indices enumerate the columns. Such a matrix can be obtained from a MATLAB `reshape` function

so that $Z_j = \texttt{reshape}(\underline{z}, \prod_{s=1}^{j} n_s, \prod_{s=j+1}^{d} n_s)$. It was shown in [70] that for a given tensor $\underline{z}$, the best rank $\{\kappa_j\}_{j=1}^{d-1}$ approximation $\tilde{\underline{z}}^*$ in the TT format always exists, and the TT approximation $\tilde{\underline{z}}$ computed from the TT-SVD algorithm is quasi-optimal, satisfying

$$\|\underline{z} - \tilde{\underline{z}}\|_F \leq \sqrt{d-1}\|\underline{z} - \tilde{\underline{z}}^*\|_F. \tag{2.32}$$

Chapter 3: Low-rank multigrid for the stochastic diffusion problem

3.1 Introduction

In this chapter we study iterative solvers for stochastic linear elliptic PDEs. As discussed in chapter 1, stochastic Galerkin discretization gives rise to large linear systems of equations which are computationally expensive to solve. These systems are in general sparse and structured. In particular, the coefficient matrix can often be expressed as a sum of tensor products of smaller matrices [65, 74]. For such systems it is natural to use an iterative solver where the coefficient matrix is never explicitly formed and matrix-vector products are computed efficiently. One way to further reduce costs is to construct low-rank approximations to the desired solution. The iterates are truncated so that the solution method handles only low-rank objects in each iteration. This idea has been used to reduce the costs of iterative solution algorithms based on Krylov subspaces. For example, a low-rank conjugate gradient method was given in [56], and low-rank GMRES methods have been studied in [4,60]. Also, a geometric multigrid method for tensor structured linear systems in high spatial dimensions was briefly discussed in [41].

We propose a low-rank multigrid method for solving the linear systems obtained from the stochastic Galerkin discretization. We consider a steady-state diffu-

sion equation with random diffusion coefficient as model problem. The linear system has Kronecker product structure and moreover, quantities used in the computation, such as the solution sought, can be expressed in matrix format. It has been shown that such systems admit low-rank approximate solutions [7, 56]. In our proposed multigrid solver, the iterates are represented as low-rank matrices, and a truncation operation is applied in each iteration to compress the matrix ranks. We derive an analytic bound for the error of the solution and show the convergence of the algorithm. We note that a convergence analysis for an iterative fixed-point like process with truncation was studied in [42]. We demonstrate using benchmark problems that the low-rank multigrid solver is often more efficient than a solver that does not use truncation, and that it is especially advantageous in reducing computing time for large-scale problems.

The material in this chapter is adapted from our published work [27]. An outline of this chapter is as follows. In section 3.2 we state the problem and briefly review the stochastic Galerkin method for the stochastic diffusion problem and the multigrid solver from which the new technique is derived. In section 3.3 we discuss the idea of low-rank approximation and introduce the multigrid solver with low-rank truncation. A convergence analysis of the low-rank multigrid solver is also given in this section. The results of numerical experiments are shown in section 3.4 to test the performance of the algorithm, and some conclusions are drawn in the last section.

3.2 Model problem

Consider the stochastic steady-state diffusion equation with homogeneous Dirichlet boundary conditions

$$\begin{cases} -\nabla \cdot (a(x,\omega)\nabla u(x,\omega)) = f(x) & \text{in } \mathcal{D} \times \Omega, \\ u(x,\omega) = 0 & \text{on } \partial\mathcal{D} \times \Omega. \end{cases} \tag{3.1}$$

Here $\mathcal{D}$ is a spatial domain and Ω is a sample space with σ-algebra $\mathcal{F}$ and probability measure P. The diffusion coefficient $a(x,\omega) : \mathcal{D} \times \Omega \to \mathbb{R}$ is a random field. We consider the case where the source term f is deterministic. The stochastic Galerkin discretization of eq. (3.1) uses a weak formulation: find $u(x,\omega) \in \mathbb{V} = H_0^1(\mathcal{D}) \otimes L^2(\Omega)$ satisfying

$$\int_\Omega \int_\mathcal{D} a(x,\omega)\nabla u(x,\omega) \cdot \nabla v(x,\omega) \mathrm{d}x \mathrm{d}P = \int_\Omega \int_\mathcal{D} f(x) v(x,\omega) \mathrm{d}x \mathrm{d}P \tag{3.2}$$

for all $v(x,\omega) \in \mathbb{V}$. The problem is well posed if $a(x,\omega)$ is bounded and strictly positive, i.e.,

$$0 < a_1 \leq a(x,\omega) \leq a_2 < \infty, \text{ a.e. } \forall x \in \mathcal{D}, \tag{3.3}$$

so that the Lax–Milgram lemma establishes existence and uniqueness of the weak solution.

We will assume that the stochastic coefficient $c(x,\omega)$ is represented as a truncated KL expansion, in terms of a finite collection of uncorrelated random variables $\{\xi_l\}_{l=1}^m$:

$$a(x,\omega) \approx a_0(x) + \sum_{l=1}^{m} \sqrt{\beta_l} a_l(x) \xi_l(\omega) \tag{3.4}$$

where $a_0(x)$ is the mean function, $(\beta_l, a_l(x))$ is the lth eigenpair of the covariance function $c(x, y)$, and the eigenvalues $\{\beta_l\}$ are assumed to be in non-increasing order. In section 3.4 we will further assume these random variables are independent and identically distributed. Let $\rho(\xi)$ be the joint density function and Γ be the joint image of $\{\xi_l\}_{l=1}^m$. The weak form of eq. (3.1) is then given as follows: find $u(x, \xi) \in \mathbb{W} = H_0^1(\mathcal{D}) \otimes L^2(\Gamma)$ s.t.

$$\int_\Gamma \rho(\xi) \int_\mathcal{D} c(x, \xi) \nabla u(x, \xi) \cdot \nabla v(x, \xi) \mathrm{d}x \mathrm{d}\xi = \int_\Gamma \rho(\xi) \int_\mathcal{D} f(x) v(x, \xi) \mathrm{d}x \mathrm{d}\xi \tag{3.5}$$

for all $v(x, \xi) \in \mathbb{W}$.

3.2.1 Stochastic Galerkin method

We briefly review the stochastic Galerkin method as described in [3, 34]. This method approximates the weak solution of eq. (3.1) in a finite-dimensional subspace

$$\mathbb{W}^{hp} = S^h \otimes T^p = \mathrm{span}\{\phi_j(x)\psi_s(\xi) \mid \phi_j(x) \in S^h, \psi_s(\xi) \in T^p\}, \tag{3.6}$$

where S^h and T^p are finite-dimensional subspaces of $H_0^1(\mathcal{D})$ and $L^2(\Gamma)$. We will use quadrilateral elements and piecewise bilinear basis functions $\{\phi_j(x)\}_{j=1}^{n_x}$ for the discretization of the physical space $H_0^1(\mathcal{D})$, and generalized polynomial chaos [96] for the stochastic basis functions $\{\psi_s(\xi)\}_{s=1}^{n_\xi}$. The latter are m-dimensional orthogonal polynomials whose total degree does not exceed p. For instance, Legendre polynomials are used if the random variables have uniform distribution with zero mean and unit variance. The number of degrees of freedom in T^p is $n_\xi = (m + p)!/(m!p!)$.

Given the subspace, now one can write the stochastic Galerkin solution as a

linear combination of the basis functions,

$$u_h(x,\xi) = \sum_{j=1}^{n_x}\sum_{s=1}^{n_\xi} u_{js}\phi_j(x)\psi_s(\xi), \tag{3.7}$$

where n_x is the dimension of the subspace S^h. Substituting eqs. (3.4) and (3.7) into eq. (3.5), and taking the test function as any basis function $\phi_i(x)\psi_r(\xi)$ results in the Galerkin system: find $\mathbf{u} \in \mathbb{R}^{n_x n_\xi}$, s.t.

$$A\mathbf{u} = \mathbf{f}. \tag{3.8}$$

The coefficient matrix A can be represented in Kronecker product notation [74],

$$A = G_0 \otimes K_0 + \sum_{l=1}^{m} G_l \otimes K_l, \tag{3.9}$$

where $\{K_l\}_{l=0}^m$ are the stiffness matrices and $\{G_l\}_{l=0}^m$ correspond to the stochastic part, with entries

$$\begin{aligned} G_0(r,s) &= \int_\Gamma \psi_r(\xi)\psi_s(\xi)\rho(\xi)\mathrm{d}\xi,\ K_0(i,j) = \int_{\mathcal{D}} a_0(x)\nabla\phi_i(x)\nabla\phi_j(x)\mathrm{d}x, \\ G_l(r,s) &= \int_\Gamma \xi_l\psi_r(\xi)\psi_s(\xi)\rho(\xi)\mathrm{d}\xi,\ K_l(i,j) = \int_{\mathcal{D}} \sqrt{\beta_l}a_l(x)\nabla\phi_i(x)\nabla\phi_j(x)\mathrm{d}x, \end{aligned} \tag{3.10}$$

$l = 1,\ldots,m;\ r,s = 1,\ldots,n_\xi;\ i,j = 1,\ldots,n_x$. The right-hand side can be written as a Kronecker product of two vectors:

$$\mathbf{f} = g_0 \otimes f_0, \tag{3.11}$$

where

$$\begin{aligned} g_0(r) &= \int_\Gamma \psi_r(\xi)\rho(\xi)\mathrm{d}\xi,\ r = 1,\ldots,n_\xi, \\ f_0(i) &= \int_{\mathcal{D}} f(x)\phi_i(x)\mathrm{d}x,\ i = 1,\ldots,n_x. \end{aligned} \tag{3.12}$$

Note that in the Galerkin system eq. (3.8), the matrix A is symmetric and positive definite. It is also blockwise sparse (see fig. 3.1) due to the orthogonality

of $\{\psi_r(\xi)\}$. The size of the linear system is in general very large ($n_x n_\xi \times n_x n_\xi$). For such a system it is suitable to use an iterative solver. Multigrid methods are among the most effective iterative solvers for the solution of discretized elliptic PDEs, capable of achieving convergence rates that are independent of the mesh size, with computational work growing only linearly with the problem size [40, 78].

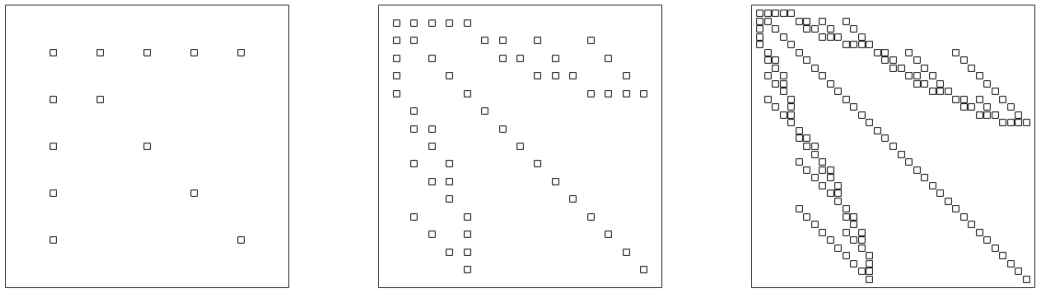


Figure 3.1: Block structure of A. $m = 4$, $p = 1, 2, 3$ from left to right. Block size is $n_x \times n_x$.

3.2.2 Multigrid

In this subsection we discuss a geometric multigrid solver proposed in [23] for the solution of the stochastic Galerkin system eq. (3.8). For this method, the mesh size h varies for different grid levels, while the polynomial degree p is held constant, i.e., the fine grid space and coarse grid space are defined as

$$\mathbb{W}^{hp} = S^h \otimes T^p, \quad \mathbb{W}^{2h,p} = S^{2h} \otimes T^p, \tag{3.13}$$

respectively. Then the prolongation and restriction operators are of the form

$$\mathscr{P} = I \otimes P, \quad \mathscr{R} = I \otimes P^T, \tag{3.14}$$

where P is the same prolongation matrix as in the deterministic case. On the coarse grid we only need to construct matrices $\{K_l^{2h}\}_{l=0}^m$, and

$$A^{2h} = G_0 \otimes K_0^{2h} + \sum_{l=1}^{m} G_l \otimes K_l^{2h}. \tag{3.15}$$

The matrices $\{G_l\}_{l=0}^m$ are the same for all grid levels.

Algorithm 3.1 describes the complete multigrid method. In each iteration, we apply one multigrid cycle (VCYCLE) for the residual equation

$$A\mathbf{c}^{(i)} = \mathbf{r}^{(i)} = \mathbf{f} - A\mathbf{u}^{(i)} \tag{3.16}$$

and update the solution $\mathbf{u}^{(i)}$ and residual $\mathbf{r}^{(i)}$. The VCYCLE function is called recursively. On the coarsest grid level ($h = h_0$) we form matrix A and solve the linear system directly. The system is of order $O(n_\xi)$ since $A \in \mathbb{R}^{n_x n_\xi \times n_x n_\xi}$ where n_x is a very small number on the coarsest grid. The smoothing function (SMOOTH) is based on a matrix splitting $A = Q - Z$ and stationary iteration

$$\mathbf{u}_{s+1} = \mathbf{u}_s + Q^{-1}(\mathbf{f} - A\mathbf{u}_s), \tag{3.17}$$

which we assume is convergent, i.e., the spectral radius $\rho(I - Q^{-1}A) < 1$. The algorithm is run until the specified relative tolerance *tol* or maximum number of iterations *maxit* is reached. It is shown in [23] that for $f \in L^2(\mathcal{D})$, the convergence rate of this algorithm is independent of the mesh size h, the number of random variables m, and the polynomial degree p.

Algorithm 3.1: Multigrid for stochastic Galerkin systems

initialization: $i = 0$, $\mathbf{r}^{(0)} = \mathbf{f}$, $r_0 = \|\mathbf{f}\|_2$
while $r > tol * r_0$ & $i \leq maxit$ **do**
    $\mathbf{c}^{(i)} = \text{VCYCLE}(A, \mathbf{0}, \mathbf{r}^{(i)})$
    $\mathbf{u}^{(i+1)} = \mathbf{u}^{(i)} + \mathbf{c}^{(i)}$
    $\mathbf{r}^{(i+1)} = \mathbf{f} - A\mathbf{u}^{(i+1)}$
    $r = \|\mathbf{r}^{(i+1)}\|_2$, $i = i + 1$
end
function $\mathbf{u}^h = \text{VCYCLE}(A^h, \mathbf{u}_0^h, \mathbf{f}^h)$
    if $h == h_0$ **then**
        solve $A^h\mathbf{u}^h = \mathbf{f}^h$ directly
    else
        $\mathbf{u}^h = \text{SMOOTH}(A^h, \mathbf{u}_0^h, \mathbf{f}^h)$
        $\mathbf{r}^h = \mathbf{f}^h - A^h\mathbf{u}^h$
        $\mathbf{r}^{2h} = \mathscr{R}\mathbf{r}^h$
        $\mathbf{c}^{2h} = \text{VCYCLE}(A^{2h}, \mathbf{0}, \mathbf{r}^{2h})$
        $\mathbf{u}^h = \mathbf{u}^h + \mathscr{P}\mathbf{c}^{2h}$
        $\mathbf{u}^h = \text{SMOOTH}(A^h, \mathbf{u}^h, \mathbf{f}^h)$
    end
end
function $\mathbf{u} = \text{SMOOTH}(A, \mathbf{u}, \mathbf{f})$
    for ν *steps* **do**
        $\mathbf{u} = \mathbf{u} + Q^{-1}(\mathbf{f} - A\mathbf{u})$
    end
end

3.3 Low-rank approximation

In this section we consider a technique designed to reduce computational effort, in terms of both time and memory use, using low-rank methods. We begin with the observation that the solution vector of the Galerkin system eq. (3.8)

$$\mathbf{u} = [u_{11}, u_{21}, \ldots, u_{n_x 1}, \ldots, u_{1n_\xi}, u_{2n_\xi}, \ldots, u_{n_x n_\xi}]^T \in \mathbb{R}^{n_x n_\xi} \tag{3.18}$$

can be restructured as a matrix

$$U = \mathrm{mat}(\mathbf{u}) = \begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1n_\xi} \\ u_{21} & u_{22} & \cdots & u_{2n_\xi} \\ \vdots & \vdots & \ddots & \vdots \\ u_{n_x 1} & u_{n_x 2} & \cdots & u_{n_x n_\xi} \end{pmatrix} \in \mathbb{R}^{n_x \times n_\xi}. \tag{3.19}$$

Then (3.8) is equivalent to a system in matrix format,

$$\mathscr{A}(U) = F, \tag{3.20}$$

where

$$\begin{aligned} \mathscr{A}(U) &= K_0 U G_0^T + \sum_{l=1}^{m} K_l U G_l^T, \\ F &= \mathrm{mat}(\mathbf{f}) = \mathrm{mat}(g_0 \otimes f_0) = f_0 g_0^T. \end{aligned} \tag{3.21}$$

It has been shown in [7, 56] that the "matricized" version of the solution U can be well approximated by a low-rank matrix when $n_x n_\xi$ is large. Evidence of this can be seen in fig. 3.2, which shows the singular values of the exact solution U for the benchmark problem discussed in section 3.4. In particular, the singular values decay exponentially, and low-rank approximate solutions can be obtained

by dropping terms from the singular value decomposition corresponding to small singular values.

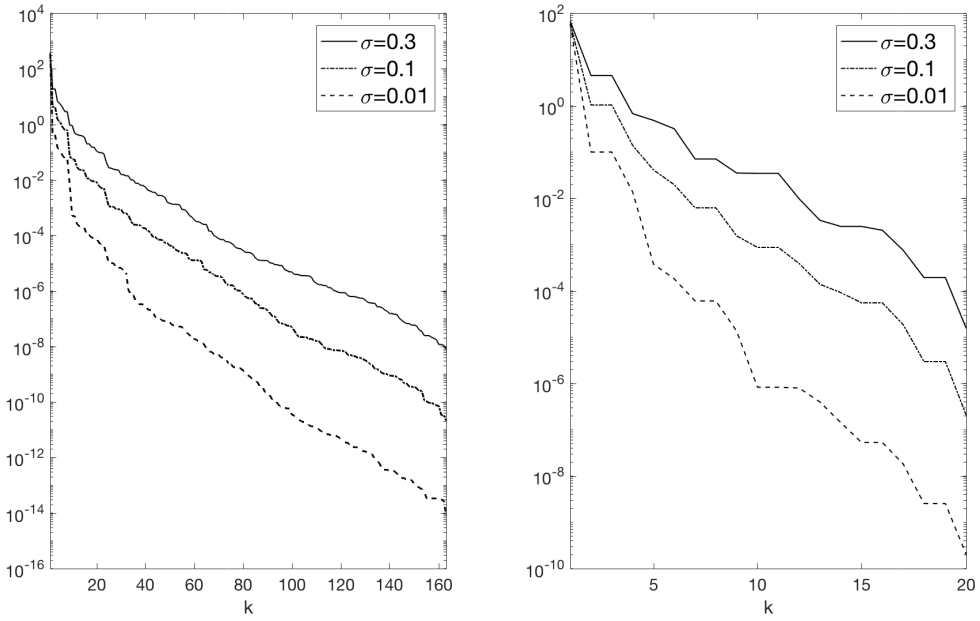


Figure 3.2: Decay of singular values of solution matrix U. Left: exponential covariance, $b = 5$, $h = 2^{-6}$, $m = 8$, $p = 3$. Right: squared exponential covariance, $b = 2$, $h = 2^{-6}$, $m = 3$, $p = 3$. See the benchmark problems in section 3.4.

Now we use low-rank approximation in the multigrid solver for eq. (3.20). Let $U^{(i)} = \mathrm{mat}(\mathbf{u}^{(i)})$ be the ith iterate, expressed in matricized format. (In the sequel, we use $\mathbf{u}^{(i)}$ and $U^{(i)}$ interchangeably to represent the equivalent vectorized or matricized quantities.) Suppose $U^{(i)}$ is represented as the outer product of two rank-κ matrices, i.e., $U^{(i)} \approx V^{(i)}W^{(i)T}$, where $V^{(i)} \in \mathbb{R}^{n_x \times \kappa}$, $W^{(i)} \in \mathbb{R}^{n_\xi \times \kappa}$. This factored form is convenient for implementation and can be readily used in basic matrix operations. For instance, the sum of two matrices gives

$$V_1^{(i)}W_1^{(i)T} + V_2^{(i)}W_2^{(i)T} = [V_1^{(i)}, V_2^{(i)}][W_1^{(i)}, W_2^{(i)}]^T. \tag{3.22}$$

Similarly, $\mathscr{A}(V^{(i)}W^{(i)T})$ can also be written as an outer product of two matrices:

$$\begin{aligned}\mathscr{A}(V^{(i)}W^{(i)T}) &= (K_0V^{(i)})(G_0W^{(i)})^T + \sum_{l=1}^{m}(K_lV^{(i)})(G_lW^{(i)})^T \\ &= [K_0V^{(i)}, K_1V^{(i)}, \ldots, K_mV^{(i)}][G_0W^{(i)}, G_1W^{(i)}, \ldots, G_mW^{(i)}]^T.\end{aligned} \tag{3.23}$$

If $V^{(i)}, W^{(i)}$ are used to represent iterates in the multigrid solver and $\kappa \ll \min(n_x, n_\xi)$, then both memory and computational (matrix-vector products) costs can be reduced, from $O(n_x n_\xi)$ to $O((n_x + n_\xi)\kappa)$. Note, however, that the ranks of the iterates may grow due to matrix additions. For example, in eq. (3.23) the rank may increase from κ to $(m+1)\kappa$ in the worst case. A way to prevent this from happening, and also to keep costs low, is to truncate the iterates and force their ranks to remain low.

3.3.1 Low-rank truncation

Our truncation strategy is derived using an idea from [56]. Assume $\tilde{X} = \tilde{V}\tilde{W}^T$, $\tilde{V} \in \mathbb{R}^{n_x \times \tilde{\kappa}}$, $\tilde{W} \in \mathbb{R}^{n_\xi \times \tilde{\kappa}}$, and $X = \mathcal{T}(\tilde{X})$ is truncated to rank κ with $X = VW^T$, $V \in \mathbb{R}^{n_x \times \kappa}$, $W \in \mathbb{R}^{n_\xi \times \kappa}$ and $\kappa < \tilde{\kappa}$. First, compute the QR factorization for both $\tilde{V}$ and $\tilde{W}$,

$$\tilde{V} = Q_{\tilde{V}}R_{\tilde{V}}, \quad \tilde{W} = Q_{\tilde{W}}R_{\tilde{W}}, \quad \text{so } \tilde{X} = Q_{\tilde{V}}R_{\tilde{V}}R_{\tilde{W}}^T Q_{\tilde{W}}^T. \tag{3.24}$$

The matrices $R_{\tilde{V}}$ and $R_{\tilde{W}}$ are of size $\tilde{\kappa} \times \tilde{\kappa}$. Next, compute an SVD of the small matrix $R_{\tilde{V}}R_{\tilde{W}}^T$:

$$R_{\tilde{V}}R_{\tilde{W}}^T = \hat{V}\,\mathrm{diag}(\sigma_1, \ldots, \sigma_{\tilde{\kappa}})\hat{W}^T \tag{3.25}$$

where $\sigma_1, \ldots, \sigma_{\tilde{\kappa}}$ are the singular values in descending order. We can truncate to a rank-κ matrix where κ is specified using either a relative criterion for singular

values,

$$\sqrt{\sigma_{\kappa+1}^2 + \cdots + \sigma_{\tilde{\kappa}}^2} \leq \epsilon_{\text{rel}} \sqrt{\sigma_1^2 + \cdots + \sigma_{\tilde{\kappa}}^2} \tag{3.26}$$

or an absolute one,

$$\kappa = \max\{\kappa \mid \sigma_k \geq \epsilon_{\text{abs}}\}. \tag{3.27}$$

Then the truncated matrices can be written in MATLAB notation as

$$V = Q_{\tilde{V}} \hat{V}(:, 1:\kappa), \quad W = Q_{\tilde{W}} \hat{W}(:, 1:\kappa) \text{diag}(\sigma_1, \ldots, \sigma_\kappa). \tag{3.28}$$

Note that the low-rank matrices X obtained from eq. (3.26) and eq. (3.27) satisfy

$$\|X - \tilde{X}\|_F \leq \epsilon_{\text{rel}} \|\tilde{X}\|_F \tag{3.29}$$

and

$$\|X - \tilde{X}\|_F \leq \epsilon_{\text{abs}} \sqrt{\tilde{\kappa} - \kappa}, \tag{3.30}$$

respectively. The right-hand side of eq. (3.30) is bounded by $\sqrt{qn_\xi}\epsilon_{\text{abs}}$, $q \leq m+2$, since in the worst case, there is a sum of $m+2$ matrices (see Line 13 of algorithm 3.2), and in general $qn_\xi < n_x$. The total cost of this computation is $O((n_x + n_\xi + \tilde{\kappa})\tilde{\kappa}^2)$. In the case where $\tilde{\kappa}$ becomes larger than n_ξ, we compute instead a direct SVD for $\tilde{X}$, which requires a matrix-matrix product to compute $\tilde{X}$ and an SVD, with smaller total cost $O(n_x n_\xi \tilde{\kappa} + n_x n_\xi^2)$.

3.3.2 Low-rank multigrid

The multigrid solver with low-rank truncation is given in algorithm 3.2. It uses truncation operators $\mathcal{T}_{\text{rel}}$ and $\mathcal{T}_{\text{abs}}$, which are defined using a relative and an absolute criterion, respectively. In each iteration, one multigrid cycle (VCYCLE) is

applied to the residual equation. Since the overall magnitudes of the singular values of the correction matrix $C^{(i)}$ decrease as $U^{(i)}$ converges to the exact solution (see fig. 3.3a for example), it is suitable to use a relative truncation tolerance ϵ_{rel} inside the VCYCLE function. It is also shown in fig. 3.3b that in each multigrid iteration, the singular values for the correction matrices C^{2h} at grids at all levels decay in a similar manner. In the smoothing function (SMOOTH), the iterate is truncated after each smoothing step using a relative criterion

$$\|\mathcal{T}_{\text{rel}_1}(U) - U\|_F \leq \epsilon_{\text{rel}} \|F^h - \mathscr{A}^h(U_0^h)\|_F \tag{3.31}$$

where A^h, U_0^h, and F^h are arguments of the VCYCLE function, and $F^h - \mathscr{A}^h(U_0^h)$ is the residual at the beginning of each V-cycle. In Line 13, the residual is truncated via a more stringent relative criterion

$$\|\mathcal{T}_{\text{rel}_2}(R^h) - R^h\|_F \leq \epsilon_{\text{rel}} h \|F^h - \mathscr{A}^h(U_0^h)\|_F \tag{3.32}$$

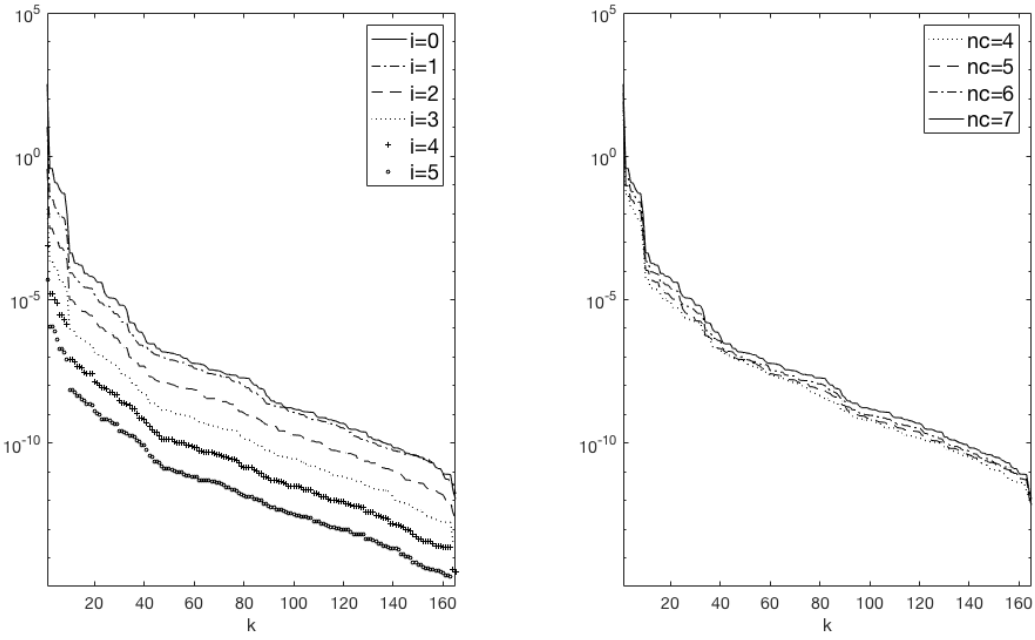
where h is the mesh size. In the main **while** loop, an absolute truncation criterion eq. (3.27) with tolerance ϵ_{abs} is used and all the singular values of $U^{(i)}$ below ϵ_{abs} are dropped. The algorithm is terminated either when the largest singular value of the residual matrix $R^{(i)}$ is smaller than ϵ_{abs} or when the multigrid solution reaches the specified accuracy (see eq. (3.62)).

Note that the post-smoothing is not explicitly required in algorithms 3.1 and 3.2, and we include it just for sake of completeness. Also, in algorithm 3.2, if the smoothing operator has the form $\mathscr{S} = S_1 \otimes S_2$, then for any matrix with a low-rank factorization $X = VW^T$, application of the smoothing operator gives

$$\mathscr{S}(X) = \mathscr{S}(VW^T) = (S_2 V)(S_1 W)^T, \tag{3.33}$$

Algorithm 3.2: Low-rank multigrid method

initialization: $i = 0$, $R^{(0)} = F$ in low-rank format, $r_0 = \|F\|_F$
while $r > tol * r_0$ & $i \leq maxit$ **do**
    $C^{(i)} = \text{VCYCLE}(A, 0, R^{(i)})$
    $\tilde{U}^{(i+1)} = U^{(i)} + C^{(i)}$, $\quad U^{(i+1)} = \mathcal{T}_{\text{abs}}(\tilde{U}^{(i+1)})$
    $\tilde{R}^{(i+1)} = F - \mathscr{A}(U^{(i+1)})$, $\quad R^{(i+1)} = \mathcal{T}_{\text{abs}}(\tilde{R}^{(i+1)})$
    $r = \|R^{(i+1)}\|_F$, $i = i + 1$
end
function $U^h = \text{VCYCLE}(A^h, U_0^h, F^h)$
    if $h == h_0$ **then**
        solve $\mathscr{A}^h(U^h) = F^h$ directly
    else
        $U^h = \text{SMOOTH}(A^h, U_0^h, F^h)$
        $\tilde{R}^h = F^h - \mathscr{A}^h(U^h)$, $\quad R^h = \mathcal{T}_{\text{rel}_2}(\tilde{R}^h)$
        $R^{2h} = \mathscr{R}(R^h)$
        $C^{2h} = \text{VCYCLE}(A^{2h}, 0, R^{2h})$
        $U^h = U^h + \mathscr{P}(C^{2h})$
        $U^h = \text{SMOOTH}(A^h, U^h, F^h)$
    end
end
function $U = \text{SMOOTH}(A, U, F)$
    for ν *steps* **do**
        $\tilde{U} = U + \mathscr{S}(F - \mathscr{A}(U))$, $\quad U = \mathcal{T}_{\text{rel}_1}(\tilde{U})$
    end
end



(a) (b)

Figure 3.3: (a) Singular values of the coarse-grid correction matrix $C^{(i)}$ at multigrid iteration $i = 0, 1, \ldots, 5$. (b) Singular values of correction matrices C^{2h} in the first multigrid iteration at various grid-refinement levels, for grid sizes $h = 2/2^{nc}, nc = 4, 5, 6, 7$. No truncation is introduced, $\sigma = 0.01$, $b = 5$, $h = 2^{-6}$, $m = 8$, $p = 3$. See the benchmark problem in section 3.4.1.

so that the result is again the outer product of two matrices of the same low rank. The prolongation and restriction operators eq. (3.14) are implemented in a similar manner. Thus, the smoothing and grid-transfer operators do not affect the ranks of matricized quantities in algorithm 3.2.

3.3.3 Convergence analysis

In order to show that algorithm 3.2 is convergent, we need to know how truncation affects the contraction of error. Consider the case of a two-grid algorithm for the linear system $A\mathbf{u} = \mathbf{f}$, where the coarse-grid solve is exact and no post-smoothing is done. Let $\bar{A}$ be the coefficient matrix on the coarse grid, let $\mathbf{e}^{(i)} = \mathbf{u} - \mathbf{u}^{(i)}$ be the error associated with $\mathbf{u}^{(i)}$, and let $\mathbf{r}^{(i)} = \mathbf{f} - A\mathbf{u}^{(i)} = A\mathbf{e}^{(i)}$ be the residual. It is shown in [23] that if no truncation is done, the error after a two-grid cycle becomes

$$\mathbf{e}^{(i+1)}_{\text{notrunc}} = (A^{-1} - \mathscr{P}\bar{A}^{-1}\mathscr{R})A(I - Q^{-1}A)^{\nu}\mathbf{e}^{(i)}, \tag{3.34}$$

and

$$\|\mathbf{e}^{(i+1)}_{\text{notrunc}}\|_A \leq C\eta(\nu)\|\mathbf{e}^{(i)}\|_A, \tag{3.35}$$

where ν is the number of pre-smoothing steps, C is a constant, and $\eta(\nu) \to 0$ as $\nu \to \infty$. The proof consists of establishing the smoothing property

$$\|A(I - Q^{-1}A)^{\nu}\mathbf{y}\|_2 \leq \eta(\nu)\|\mathbf{y}\|_A, \quad \forall \mathbf{y} \in \mathbb{R}^{n_x n_\xi}, \tag{3.36}$$

and the approximation property

$$\|(A^{-1} - \mathscr{P}\bar{A}^{-1}\mathscr{R})\mathbf{y}\|_A \leq C\|\mathbf{y}\|_2, \quad \forall \mathbf{y} \in \mathbb{R}^{n_x n_\xi}, \tag{3.37}$$

and applying these bounds to eq. (3.34).

Now we derive an error bound for algorithm 3.2. The result is presented in two steps. First, we consider the VCYCLE function only; the following lemma shows the effect of the relative truncations defined in eqs. (3.31) and (3.32).

Lemma 3.1. *Let* $\mathbf{u}^{(i+1)} = \text{VCYCLE}(A, \mathbf{u}^{(i)}, \mathbf{f})$ *and let* $\mathbf{e}^{(i+1)} = \mathbf{u} - \mathbf{u}^{(i+1)}$ *be the associated error. Assume a damped Jacobi smoother is used (see eq. (3.69)). Then*

$$\|\mathbf{e}^{(i+1)}\|_A \leq C_1(\nu)\|\mathbf{e}^{(i)}\|_A, \tag{3.38}$$

where, for small enough ϵ_{rel} *and large enough* ν, $C_1(\nu) < 1$ *independent of the mesh size* h.

Proof. For $s = 1, \ldots, \nu$, let $\tilde{\mathbf{u}}_s^{(i)}$ be the quantity computed after application of the smoothing operator at step s before truncation, and let $\mathbf{u}_s^{(i)}$ be the modification obtained from truncation by $\mathcal{T}_{\text{rel}_1}$ of eq. (3.31). For example,

$$\tilde{\mathbf{u}}_1^{(i)} = \mathbf{u}^{(i)} + Q^{-1}(\mathbf{f} - A\mathbf{u}^{(i)}), \quad \mathbf{u}_1^{(i)} = \mathcal{T}_{\text{rel}_1}(\tilde{\mathbf{u}}_1^{(i)}). \tag{3.39}$$

Denote the associated error as $\mathbf{e}_s^{(i)} = \mathbf{u} - \mathbf{u}_s^{(i)}$. From eq. (3.31), we have

$$\mathbf{e}_1^{(i)} = (I - Q^{-1}A)\mathbf{e}^{(i)} + \delta_1^{(i)}, \quad \text{where } \|\delta_1^{(i)}\|_2 \leq \epsilon_{\text{rel}}\|\mathbf{r}^{(i)}\|_2. \tag{3.40}$$

Similarly, after ν smoothing steps,

$$\begin{aligned} \mathbf{e}_\nu^{(i)} &= (I - Q^{-1}A)^\nu \mathbf{e}^{(i)} + \Delta_\nu^{(i)} \\ &= (I - Q^{-1}A)^\nu \mathbf{e}^{(i)} + (I - Q^{-1}A)^{\nu-1}\delta_1^{(i)} + \cdots + (I - Q^{-1}A)\delta_{\nu-1}^{(i)} + \delta_\nu^{(i)}, \end{aligned} \tag{3.41}$$

where

$$\|\delta_s^{(i)}\|_2 \leq \epsilon_{\text{rel}}\|\mathbf{r}^{(i)}\|_2, \quad s = 1, \ldots, \nu. \tag{3.42}$$

In Line 13 of algorithm 3.2, the residual $\tilde{\mathbf{r}}_\nu^{(i)} = A\mathbf{e}_\nu^{(i)}$ is truncated to $\mathbf{r}_\nu^{(i)}$ via eq. (3.32), so that

$$\|\mathbf{r}_\nu^{(i)} - \tilde{\mathbf{r}}_\nu^{(i)}\|_2 \leq \epsilon_{\text{rel}} h \|\mathbf{r}^{(i)}\|_2. \tag{3.43}$$

Let $\tau^{(i)} = \mathbf{r}_\nu^{(i)} - \tilde{\mathbf{r}}_\nu^{(i)}$. Referring to eqs. (3.34) and (3.41), we can write the error associated with $\mathbf{u}^{(i+1)}$ as

$$\begin{aligned}
\mathbf{e}^{(i+1)} &= \mathbf{e}_\nu^{(i)} - \mathscr{P}\bar{A}^{-1}\mathscr{R}\mathbf{r}_\nu^{(i)} \\
&= (I - \mathscr{P}\bar{A}^{-1}\mathscr{R}A)\mathbf{e}_\nu^{(i)} - \mathscr{P}\bar{A}^{-1}\mathscr{R}\tau^{(i)} \\
&= \mathbf{e}_{\text{notrunc}}^{(i+1)} + (A^{-1} - \mathscr{P}\bar{A}^{-1}\mathscr{R})A\Delta_\nu^{(i)} - \mathscr{P}\bar{A}^{-1}\mathscr{R}\tau^{(i)} \\
&= \mathbf{e}_{\text{notrunc}}^{(i+1)} + (A^{-1} - \mathscr{P}\bar{A}^{-1}\mathscr{R})(A\Delta_\nu^{(i)} + \tau^{(i)}) - A^{-1}\tau^{(i)}.
\end{aligned} \tag{3.44}$$

Applying the approximation property eq. (3.37) gives

$$\|(A^{-1} - \mathscr{P}\bar{A}^{-1}\mathscr{R})(A\Delta_\nu^{(i)} + \tau^{(i)})\|_A \leq C(\|A\Delta_\nu^{(i)}\|_2 + \|\tau^{(i)}\|_2). \tag{3.45}$$

Using the fact that for any matrix $B \in \mathbb{R}^{n_x n_\xi \times n_x n_\xi}$,

$$\sup_{\mathbf{y}\neq\mathbf{0}} \frac{\|B\mathbf{y}\|_A}{\|\mathbf{y}\|_A} = \sup_{\mathbf{y}\neq\mathbf{0}} \frac{\|A^{1/2}B\mathbf{y}\|_2}{\|A^{1/2}\mathbf{y}\|_2} = \sup_{\mathbf{z}\neq\mathbf{0}} \frac{\|A^{1/2}BA^{-1/2}\mathbf{z}\|_2}{\|\mathbf{z}\|_2} = \|A^{1/2}BA^{-1/2}\|_2, \tag{3.46}$$

we get

$$\begin{aligned}
\|A(I - Q^{-1}A)^{\nu-s}\delta_s^{(i)}\|_2 &\leq \|A^{1/2}\|_2 \, \|(I - Q^{-1}A)^{\nu-s}\delta_s^{(i)}\|_A \\
&\leq \|A^{1/2}\|_2 \, \|A^{1/2}(I - Q^{-1}A)^{\nu-s}A^{-1/2}\|_2 \, \|\delta_s^{(i)}\|_A \\
&\leq \rho(I - Q^{-1}A)^{\nu-s}\|A^{1/2}\|_2^2 \, \|\delta_s^{(i)}\|_2
\end{aligned} \tag{3.47}$$

where ρ is the spectral radius. We have used the fact that $A^{1/2}(I - Q^{-1}A)^{\nu-s}A^{-1/2}$ is a symmetric matrix (since Q is symmetric). Define $d_1(\nu) = (\rho(I - Q^{-1}A)^{\nu-1} +$

$\cdots + \rho(I - Q^{-1}A) + 1)\|A^{1/2}\|_2^2$. Then eqs. (3.42) and (3.43) imply that

$$\begin{aligned} \|A\Delta_\nu^{(i)}\|_2 + \|\tau^{(i)}\|_2 &\le \epsilon_{\text{rel}}(d_1(\nu) + h)\|\mathbf{r}^{(i)}\|_2 \\ &\le \epsilon_{\text{rel}}(d_1(\nu) + h)\|A^{1/2}\|_2 \, \|\mathbf{e}^{(i)}\|_A. \end{aligned} \tag{3.48}$$

On the other hand,

$$\begin{aligned} \|A^{-1}\tau^{(i)}\|_A = (A^{-1}\tau^{(i)}, \tau^{(i)})^{1/2} &\le \|A^{-1}\|_2^{1/2} \, \|\tau^{(i)}\|_2 \\ &\le \epsilon_{\text{rel}} h\|A^{-1}\|_2^{1/2} \, \|\mathbf{r}^{(i)}\|_2 \\ &\le \epsilon_{\text{rel}} h\|A^{-1}\|_2^{1/2} \, \|A^{1/2}\|_2 \, \|\mathbf{e}^{(i)}\|_A. \end{aligned} \tag{3.49}$$

Combining eqs. (3.35), (3.44), (3.45), (3.48) and (3.49), we conclude that

$$\|\mathbf{e}^{(i+1)}\|_A \le C_1(\nu)\|\mathbf{e}^{(i)}\|_A \tag{3.50}$$

where

$$C_1(\nu) = C\eta(\nu) + \epsilon_{\text{rel}}(C(d_1(\nu) + h) + h\|A^{-1}\|_2^{1/2})\|A^{1/2}\|_2. \tag{3.51}$$

Note that $\rho(I - Q^{-1}A) < 1$, $\|A\|_2$ is bounded by a constant, and $\|A^{-1}\|_2$ is of order $O(h^{-2})$ [74]. Thus, for small enough ϵ_{rel} and large enough ν, $C_1(\nu)$ is bounded below 1 independent of h. □

Next, we adjust this argument by considering the effect of the absolute truncations in the main **while** loop. In algorithm 3.2, the VCYCLE is used for the residual equation, and the updated solution $\tilde{\mathbf{u}}^{(i+1)}$ and residual $\tilde{\mathbf{r}}^{(i+1)}$ are truncated to $\mathbf{u}^{(i+1)}$ and $\mathbf{r}^{(i+1)}$, respectively, using an absolute truncation criterion as in eq. (3.27). Thus, at the ith iteration ($i > 1$), the residual passed to the VCYCLE function is in fact a perturbed residual, i.e.,

$$\mathbf{r}^{(i)} = \tilde{\mathbf{r}}^{(i)} + \gamma = A\mathbf{e}^{(i)} + \gamma, \quad \text{where } \|\gamma\|_2 \le \sqrt{qn_\xi}\epsilon_{\text{abs}}. \tag{3.52}$$

It follows that in the first smoothing step,

$$\tilde{\mathbf{u}}_1^{(i)} = \mathbf{u}^{(i)} + Q^{-1}(\mathbf{f} - A\mathbf{u}^{(i)} + \gamma), \quad \mathbf{u}_1^{(i)} = \mathcal{T}_{\text{rel}_1}(\tilde{\mathbf{u}}_1^{(i)}), \tag{3.53}$$

and this introduces an extra term in $\Delta_\nu^{(i)}$ (see eq. (3.41)),

$$\Delta_\nu^{(i)} = (I - Q^{-1}A)^{\nu-1}\delta_1^{(i)} + \cdots + (I - Q^{-1}A)\delta_{\nu-1}^{(i)} + \delta_\nu^{(i)} - (I - Q^{-1}A)^{\nu-1}Q^{-1}\gamma. \tag{3.54}$$

As in the derivation of eq. (3.47), we have

$$\|A(I - Q^{-1}A)^{\nu-1}Q^{-1}\gamma\|_2 \le \rho(I - Q^{-1}A)^{\nu-1}\|A^{1/2}\|_2^2\,\|Q^{-1}\|_2\,\|\gamma\|_2. \tag{3.55}$$

In the case of a damped Jacobi smoother, $\|Q^{-1}\|_2$ is bounded by a constant. Denote $d_2(\nu) = \rho(I - Q^{-1}A)^{\nu-1}\|A^{1/2}\|_2^2\,\|Q^{-1}\|_2$. Also note that $\|\mathbf{r}^{(i)}\|_2 \le \|A^{1/2}\|_2\,\|\mathbf{e}^{(i)}\|_A + \|\gamma\|_2$. Then eqs. (3.48) and (3.49) are modified to

$$\begin{aligned}
&\|A\Delta_\nu^{(i)}\|_2 + \|\tau^{(i)}\|_2 \\
&\le \epsilon_{\text{rel}}(d_1(\nu) + h)\|\mathbf{r}^{(i)}\|_2 + d_2(\nu)\|\gamma\|_2 \\
&\le \epsilon_{\text{rel}}(d_1(\nu) + h)\|A^{1/2}\|_2\,\|\mathbf{e}^{(i)}\|_A + (d_2(\nu) + \epsilon_{\text{rel}}(d_1(\nu) + h))\|\gamma\|_2,
\end{aligned} \tag{3.56}$$

and

$$\|A^{-1}\tau^{(i)}\|_A \le \epsilon_{\text{rel}}h\|A^{-1}\|_2^{1/2}\,\|A^{1/2}\|_2\,\|\mathbf{e}^{(i)}\|_A + \epsilon_{\text{rel}}h\|A^{-1}\|_2^{1/2}\,\|\gamma\|_2. \tag{3.57}$$

As we truncate the updated solution $\tilde{\mathbf{u}}^{(i+1)}$, we have

$$\mathbf{u}^{(i+1)} = \tilde{\mathbf{u}}^{(i+1)} + \zeta, \quad \text{where } \|\zeta\|_2 \le \sqrt{qn_\xi}\epsilon_{\text{abs}}. \tag{3.58}$$

Let

$$C_2(\nu) = (Cd_2(\nu) + \epsilon_{\text{rel}}(C(d_1(\nu) + h) + h\|A^{-1}\|_2^{1/2}) + \|A^{1/2}\|_2)\sqrt{q}. \tag{3.59}$$

From eqs. (3.56) to (3.59), we conclude with the following theorem:

Theorem 3.2. *Let* $\mathbf{e}^{(i)} = \mathbf{u} - \mathbf{u}^{(i)}$ *denote the error at the* i*th iteration of algorithm 3.2. Then*

$$\|\mathbf{e}^{(i+1)}\|_A \leq C_1(\nu)\|\mathbf{e}^{(i)}\|_A + C_2(\nu)\sqrt{n_\xi}\epsilon_{abs}, \tag{3.60}$$

where $C_1(\nu) < 1$ *for large enough* ν *and small enough* ϵ_{rel}*, and* $C_2(\nu)$ *is bounded by a constant. Also, eq. (3.60) implies that*

$$\|\mathbf{e}^{(i)}\|_A \leq C_1^i(\nu)\|\mathbf{e}^{(0)}\|_A + \frac{1 - C_1^i(\nu)}{1 - C_1(\nu)} C_2(\nu)\sqrt{n_\xi}\epsilon_{abs}, \tag{3.61}$$

i.e., the A-norm of the error for the low-rank multigrid solution at the i*th iteration is bounded by* $C_1^i(\nu)\|\mathbf{e}^{(0)}\|_A + O(\sqrt{n_\xi}\epsilon_{abs})$*. Thus, algorithm 3.2 converges until the A-norm of the error becomes as small as* $O(\sqrt{n_\xi}\epsilon_{abs})$*.*

In the proof above, it is convenient to consider the damped Jacobi smoother in that the matrix Q is symmetric and $\|Q^{-1}\|_2$ is bounded. In fact, one can use the smoothing property eq. (3.36) to bound eq. (3.47), which does not require symmetry in Q, and the proof can be generalized for any smoother with bounded $\|Q^{-1}\|_2$. Also, it can be shown that the result in theorem 3.2 holds if post-smoothing is used. The convergence of full (recursive) multigrid with these truncation operations can be established following an inductive argument analogous to that in the deterministic case (see, e.g., [26, 40]). Besides, in algorithm 3.2, the truncation on $\tilde{\mathbf{r}}^{(i+1)}$ imposes a stopping criterion, i.e.,

$$\begin{aligned} \|\tilde{\mathbf{r}}^{(i+1)}\|_2 &\leq \|\tilde{\mathbf{r}}^{(i+1)} - \mathbf{r}^{(i+1)}\|_2 + \|\mathbf{r}^{(i+1)}\|_2 \\ &\leq \sqrt{qn_\xi}\epsilon_{\text{abs}} + tol * r_0. \end{aligned} \tag{3.62}$$

In section 3.4 we will vary the value of ϵ_{abs} and see how the low-rank multigrid solver works compared with algorithm 3.1 where no truncation is done.

Remark 3.3. It is shown in [74] that for eq. (3.8), with constant mean a_0 and standard deviation σ,

$$\|A\|_2 = \alpha(a_0 + \sigma C_{p+1}^{\max} \sum_{l=1}^{m} \sqrt{\beta_l} \|a_l(x)\|_\infty), \tag{3.63}$$

where $C_{p+1}^{\max}$ is the maximal root of an orthogonal polynomial of degree $p+1$, and α is a constant independent of h, m, and p. If Legendre polynomials on the interval $[-1, 1]$ are used, $C_{p+1}^{\max} < 1$. Since both C_1 and C_2 in theorem 3.2 are related to $\|A\|_2$, the convergence rate of algorithm 3.2 will depend on m. However, if the eigenvalues $\{\beta_l\}$ decay fast, this dependence is negligable.

Remark 3.4. As shown in eq. (3.51), the factor h in the truncation criterion eq. (3.32) is introduced to compensate for the order $O(h^{-2})$ of $\|A^{-1}\|_2$. Arguments similar to those in [74] can also be used to show that if A comes from a model where the diffusion coefficient is a lognormal random field, then $\|A^{-1}\|_2 = O(h^{-2})$ (see the discussions in [29, 92]), and the error bound in theorem 3.2 is still valid.

Remark 3.5. If instead a relative truncation is used in the **while** loop so that

$$\mathbf{r}^{(i+1)} = \tilde{\mathbf{r}}^{(i+1)} + \gamma = A\mathbf{e}^{(i+1)} + \gamma, \quad \text{where } \|\gamma\|_2 \leq \epsilon_{\text{rel}} \|\tilde{\mathbf{r}}^{(i+1)}\|_2, \tag{3.64}$$

then a similar convergence result can be derived, and the algorithm stops when

$$\|\tilde{\mathbf{r}}^{(i+1)}\|_2 \leq \frac{tol * r_0}{1 - \epsilon_{\text{rel}}}. \tag{3.65}$$

However, the relative truncation in general results in a larger rank for $\mathbf{r}^{(i)}$, and the improvement in efficiency will be less significant.

3.4 Numerical experiments

Consider the benchmark problem with a two-dimensional spatial domain $\mathcal{D} = [-1,1]^2$ and constant source term $f = 1$. We look at two different forms for the covariance function $c(x,y)$ of the diffusion coefficient $a(x,\omega)$.

3.4.1 Exponential covariance

The exponential covariance function takes the form

$$c(x,y) = \sigma^2 \exp\left(-\frac{1}{b}\|x-y\|_1\right). \tag{3.66}$$

This is a convenient choice because there are known analytic solutions for the eigenpair $(\beta_l, a_l(x))$ [34]. In the KL expansion, take $a_0(x) = 1$ and $\{\xi_l\}_{l=1}^m$ independent and uniformly distributed on $[-1,1]$:

$$a(x,\omega) = a_0(x) + \sqrt{3}\sum_{l=1}^{m}\sqrt{\beta_l}a_l(x)\xi_l(\omega). \tag{3.67}$$

Then $\sqrt{3}\xi_l$ has zero mean and unit variance, and Legendre polynomials are used as basis functions for the stochastic space. The correlation length b affects the decay of $\{\beta_l\}$ in the KL expansion. The number of random variables m is chosen so that

$$\left(\sum_{l=1}^{m}\beta_l\right) \Big/ \left(\sum_{l=1}^{M}\beta_l\right) \geq 95\%. \tag{3.68}$$

Here M is a large number which we set as 1000.

We now examine the performance of the multigrid solver with low-rank truncation. We employ a damped Jacobi smoother, with

$$Q = \frac{1}{\omega_s}\mathrm{diag}(A) = \frac{1}{\omega_s} I \otimes \mathrm{diag}(K_0) \tag{3.69}$$

(since $G_0 = I$ and $\text{diag}(G_l) = 0$ for $l = 1, \ldots, m$), and the parameter value $\omega_s = 2/3$. Apply three smoothing steps ($\nu = 3$) in the SMOOTH function. Set the multigrid $tol = 10^{-6}$. As shown in eq. (3.62), the relative residual $\|F - \mathscr{A}(U^{(i)})\|_F / \|F\|_F$ for the solution $U^{(i)}$ produced in algorithm 3.2 is related to the value of the truncation tolerance ϵ_{abs}. In all the experiments, we also run the multigrid solver without truncation to reach a relative residual that is closest to what we get from the low-rank multigrid solver. We fix the relative truncation tolerance ϵ_{rel} as 10^{-2}. (The truncation criteria in eqs. (3.31) and (3.32) are needed for the analysis. In practice we found the performance with the relative criterion in eq. (3.29) to be essentially the same as the results shown in this section.) The numerical results, i.e., the rank of multigrid solution, the number of iterations, and the elapsed time (in seconds) for solving the Galerkin system, are given in tables 3.1 to 3.3. In all the tables, the 3rd and 4th columns are the results of low-rank multigrid with different values of truncation tolerance ϵ_{abs}, and for comparison the last two columns show the results for the multigrid solver without truncation. The Galerkin systems are generated from the Incompressible Flow and Iterative Solver Software (IFISS, [82]). All computations are done in MATLAB 9.1.0 (R2016b) on a MacBook with 4 GB SDRAM.

Table 3.1 shows the performance of the multigrid solver for various mesh sizes h, or spatial degrees of freedom n_x, with other parameters fixed. The 3rd and 5th columns show that multigrid with low-rank truncation uses less time than the standard multigrid solver. This is especially true when n_x is large: for $h = 2^{-8}$, $n_x = 261121$, low-rank approximation reduces the computing time from 2857s to 370s. The improvement is much more significant (see the 4th and 6th columns) if

the problem does not require very high accuracy for the solution. Table 3.2 shows the results for various degrees of freedom n_ξ in the stochastic space. The multigrid solver with absolute truncation tolerance 10^{-6} is more efficient compared with no truncation in all cases and uses only about half the time. The 4th and 6th columns indicate that the decrease in computing time by low-rank truncation is more obvious with the larger tolerance 10^{-4}.

Table 3.1: Performance of multigrid solver with $\epsilon_{\text{abs}} = 10^{-6}$, 10^{-4}, and no truncation for various $n_x = (2/h-1)^2$. Exponential covariance, $\sigma = 0.01$, $b = 4$, $m = 11$, $p = 3$, $n_\xi = 364$.

		$\epsilon_{\text{abs}} = 10^{-6}$	$\epsilon_{\text{abs}} = 10^{-4}$	No truncation	
64×64 grid $h = 2^{-5}$ $n_x = 3969$	Rank Iterations Elapsed time Rel residual	51 5 6.26 1.51e-6	12 4 1.63 6.05e-5	 5 12.60 9.97e-7	 4 10.08 1.38e-5
128×128 grid $h = 2^{-6}$ $n_x = 16129$	Rank Iterations Elapsed time Rel residual	51 6 20.90 2.45e-6	12 4 5.17 9.85e-5	 5 54.59 1.23e-6	 3 32.92 2.20e-4
256×256 grid $h = 2^{-7}$ $n_x = 65025$	Rank Iterations Elapsed time Rel residual	49 5 76.56 4.47e-6	13 4 24.31 2.07e-4	 5 311.27 1.36e-6	 3 188.70 2.35e-04
512×512 grid $h = 2^{-8}$ $n_x = 261121$	Rank Iterations Elapsed time Rel residual	39 5 370.98 9.93e-6	16 3 86.30 4.33e-4	 4 2857.82 1.85e-5	 3 2099.06 2.43e-4

We have observed that when the standard deviation σ in the covariance function (3.66) is smaller, the singular values of the solution matrix U decay faster (see fig. 3.2), and it is more suitable for low-rank approximation. This is also shown in

Table 3.2: Performance of multigrid solver with $\epsilon_{\text{abs}} = 10^{-6}$, 10^{-4}, and no truncation for various $n_\xi = (m+p)!/(m!p!)$. Exponential covariance, $\sigma = 0.01$, $h = 2^{-6}$, $p = 3$, $n_x = 16129$.

		$\epsilon_{\text{abs}} = 10^{-6}$	$\epsilon_{\text{abs}} = 10^{-4}$	No truncation	
	Rank	25	9		
$b = 5, m = 8$	Iterations	5	4	5	3
$n_\xi = 165$	Elapsed time	5.82	1.71	19.33	11.65
	Rel residual	5.06e-6	3.41e-4	1.22e-6	2.20e-4
	Rank	51	12		
$b = 4, m = 11$	Iterations	6	4	5	3
$n_\xi = 364$	Elapsed time	20.90	5.17	54.59	32.92
	Rel residual	2.45e-6	9.85e-5	1.23e-6	2.20e-4
	Rank	91	23		
$b = 3, m = 16$	Iterations	6	5	5	4
$n_\xi = 969$	Elapsed time	97.34	16.96	197.82	158.56
	Rel residual	5.71e-7	3.99e-5	1.23e-6	1.63e-5
	Rank	165	86		
$b = 2.5, m = 22$	Iterations	6	5	6	4
$n_\xi = 2300$	Elapsed time	648.59	172.41	1033.29	682.45
	Rel residual	1.59e-7	8.57e-6	9.29e-8	1.63e-5

the numerical results. In the previous cases, we fixed σ as 0.01. In table 3.3, the advantage of low-rank multigrid is clearer for a smaller σ, and the solution is well approximated by a matrix of smaller rank. On the other hand, as the value of σ increases, the singular values of the matricized solution, as well as the matricized iterates, decay more slowly and the same truncation criterion gives higher-rank objects. Thus, the total time for solving the system and the time spent on truncation will also increase. Another observation from the above numerical experiments is that the iteration counts are largely unaffected by truncation. In algorithm 3.2, similar numbers of iterations are required to reach a comparable accuracy as in the cases with no truncation.

Table 3.3: Performance of multigrid solver with $\epsilon_{\text{abs}} = 10^{-6}$, 10^{-4}, and no truncation for various σ. Time spent on truncation is given in parentheses. Exponential covariance, $b = 4$, $h = 2^{-6}$, $m = 11$, $p = 3$, $n_x = 16129$, $n_\xi = 364$.

		$\epsilon_{\text{abs}} = 10^{-6}$	$\epsilon_{\text{abs}} = 10^{-4}$	No truncation	
$\sigma = 0.001$	Rank	13	12		
	Iterations	6	4	5	4
	Elapsed time	7.61 (4.77)	3.73 (2.29)	54.43	43.58
	Rel residual	1.09e-6	6.53e-5	1.22e-6	1.63e-5
$\sigma = 0.01$	Rank	51	12		
	Iterations	6	4	5	3
	Elapsed time	20.90 (15.05)	5.17 (3.16)	54.59	32.92
	Rel residual	2.45e-6	9.85e-5	1.23e-6	2.20e-4
$\sigma = 0.1$	Rank	136	54		
	Iterations	6	4	5	3
	Elapsed time	54.44 (33.91)	18.12 (12.70)	55.49	33.62
	Rel residual	3.28e-6	2.47e-4	1.88e-6	2.62e-4
$\sigma = 0.3$	Rank	234	128		
	Iterations	9	7	8	4
	Elapsed time	138.63 (77.54)	60.96 (38.66)	86.77	43.42
	Rel residual	6.03e-6	4.71e-4	2.99e-6	7.76e-4

3.4.2 Squared exponential covariance

In the second example we consider covariance function

$$c(x,y) = \sigma^2 \exp\left(-\frac{1}{b^2}\|x-y\|_2^2\right). \tag{3.70}$$

The eigenpair $(\beta_l, a_l(x))$ is computed via a finite element approximation of the eigenvalue problem

$$\int_{\mathcal{D}} c(x,y)a_l(y)\mathrm{d}y = \beta_l a_l(x). \tag{3.71}$$

Again, in the KL expansion eq. (3.67), take $a_0(x) = 1$ and $\{\xi_l\}_{l=1}^m$ independent and uniformly distributed on $[-1,1]$. The eigenvalues of the squared exponential covariance eq. (3.70) decay much faster than those of eq. (3.66), and thus fewer terms are required to satisfy eq. (3.68). For instance, for $b = 2$, $m = 3$ will suffice. Table 3.4 shows the performance of multigrid with low-rank truncation for various spatial degrees of freedom n_x. In this case, we are able to work with finer meshes since the value of n_ξ is smaller. In all experiments the low-rank multigrid solver uses less time compared with no truncation.

3.5 Conclusions

In this chapter we focused on the multigrid solver, one of the most efficient iterative solvers, for the stochastic steady-state diffusion problem. We discussed how to combine the idea of low-rank approximation with multigrid to reduce computational costs. We proved the convergence of the low-rank multigrid method with an analytic error bound. It was shown in numerical experiments that the low-rank

Table 3.4: Performance of multigrid solver with $\epsilon_{\text{abs}} = 10^{-6}$, 10^{-4}, and no truncation for various $n_x = (2/h-1)^2$. Squared exponential covariance, $\sigma = 0.01$, $b = 2$, $m = 3$, $p = 3$, $n_\xi = 20$.

		$\epsilon_{\text{abs}} = 10^{-6}$	$\epsilon_{\text{abs}} = 10^{-4}$	No truncation	
128×128 grid $h = 2^{-6}$ $n_x = 16129$	Rank	9	4		
	Iterations	5	3	4	3
	Elapsed time	0.78	0.35	1.08	0.82
	Rel residual	1.20e-5	9.15e-4	1.63e-5	2.20e-4
256×256 grid $h = 2^{-7}$ $n_x = 65025$	Rank	8	4		
	Iterations	4	3	4	3
	Elapsed time	2.55	1.31	4.58	3.46
	Rel residual	3.99e-5	9.09e-4	1.78e-05	2.35e-4
512×512 grid $h = 2^{-8}$ $n_x = 261121$	Rank	8	2		
	Iterations	4	2	4	2
	Elapsed time	10.23	2.13	18.93	9.61
	Rel residual	6.41e-5	6.91e-3	1.85e-5	3.29e-3
1024×1024 grid $h = 2^{-9}$ $n_x = 1045629$	Rank	8	2		
	Iterations	4	2	4	2
	Elapsed time	58.09	10.66	115.75	63.54
	Rel residual	6.41e-5	6.93e-3	1.90e-5	3.32e-3

truncation is useful in decreasing the computing time when the variance of the random coefficient is relatively small. The proposed algorithm also exhibited great advantage for problems with large numbers of spatial degrees of freedom.

Chapter 4: Low-rank methods for stochastic eigenvalue problems

4.1 Introduction

In this chapter we study low-rank solution methods for stochastic eigenvalue problems associated with discrete PDE operators. Approaches for solving stochastic eigenvalue problems can be broadly divided into non-intrusive methods, including Monte Carlo methods and stochastic collocation methods [1, 76], and intrusive stochastic Galerkin methods. The Galerkin approach gives parametrized descriptions of the eigenvalues and eigenvectors, represented as expansions with stochastic basis functions. A commonly used framework is the generalized polynomial chaos (gPC) expansion [96]. A direct projection onto the subspace spanned by the basis functions will result in large coupled nonlinear systems that can be solved by a Newton-type algorithm [9, 33]. Alternatives that do not use nonlinear solvers are stochastic versions of the (inverse) power methods and subspace iteration algorithms [43, 44, 67, 86, 93]. These methods have been shown to produce accurate solutions compared with the Monte Carlo or collocation methods. However, due to the extra dimensions introduced by randomness, solving the linear systems, as well as other computations, can be expensive. In this chapter, we develop new efficient solution methods that use low-rank approximations for the stochastic eigenvalue

problems.

We use the stochastic Galerkin approach to compute gPC expansions of one or more minimal eigenvalues and corresponding eigenvectors of parameter-dependent matrices, arising from discretization of stochastic self-adjoint PDEs. Our work builds on the results in [67, 86]. We devise a low-rank variant of the stochastic inverse subspace iteration algorithm, where the iterates and solutions are approximated by low-rank matrices. In each iteration, the linear system solves required by the inverse iteration algorithm are performed by low-rank iterative solvers. The orthonormalization and Rayleigh quotient computations in the algorithm are also computed with the low-rank representation. To test the efficiency of the proposed algorithm, we consider two benchmark problems, a stochastic diffusion problem and a Schur complement operator derived from a discrete stochastic Stokes problem. The diffusion problem has some poorly separated eigenvalues and we show that a generalization of Rayleigh–Ritz refinement for the stochastic problem can be used to obtain good approximations. A low-rank geometric multigrid method is used for solving the linear systems. For the Stokes problem, the minimal eigenvalue of the Schur complement operator is the square of the parametrized inf-sup stability constant for the Stokes operator. Each step of the inverse iteration entails solving a Stokes system for which a low-rank variant of the MINRES method is used. We demonstrate the accuracy of the solutions and efficiency of the low-rank algorithms by comparison with the Monte Carlo method and the full subspace iteration algorithm without using low-rank approximation.

We note that a low-rank variant of LOBPCG method was studied in [57] for

eigenvalue problems from discretization of high-dimensional elliptic PDEs. A low-rank Arnoldi method was proposed in [10] to approximate the posterior covariance matrix in stochastic inverse problems. Another dimension reduction technique is the reduced basis method. This idea was used in [30, 47, 63], where the eigenvectors are approximated from a linear space spanned by carefully selected sample "snapshot" solutions obtained via, for instance, a greedy algorithm that minimizes an a posteriori error estimator. Inf-sup stability problems were also studied in [48, 84] in which lower and upper bounds for the smallest eigenvalue of a stochastic Hermitian matrix are computed using successive constraint methods in the reduced basis context.

The material presented in this chapter is based on our work in [28]. The rest of this chapter is organized as follows. In section 4.2 we review the stochastic inverse subspace iteration algorithm for computing several minimal eigenvalues and corresponding eigenvectors of parameter-dependent matrices. In section 4.3 we introduce the idea of low-rank approximation in this setting, and discuss how computations in the inverse subspace iteration algorithm are done efficiently with quantities in low-rank format. The stochastic diffusion problem and the stochastic Stokes problem are discussed in sections 4.4 and 4.5, respectively, with numerical results showing the effectiveness of the low-rank algorithms. Conclusions are drawn in the last section.

4.2 Stochastic inverse subspace iteration

Let $(\Omega, \mathcal{F}, P)$ be a probability triplet where Ω is a sample space with σ-algebra $\mathcal{F}$ and probability measure P. Define a random variable $\xi : \Omega \to \Gamma \subset \mathbb{R}^m$ with uncor-

related components and let μ be the induced measure on Γ. Consider the following stochastic eigenvalue problem: find n_e minimal eigenvalues $\lambda^s(\xi)$ and corresponding eigenvectors $u^s(\xi)$ such that

$$A(\xi)u^s(\xi) = \lambda^s(\xi)u^s(\xi), \quad s = 1, 2, \ldots, n_e, \tag{4.1}$$

almost surely, where $A(\xi)$ is a matrix-valued random variable. We will use a version of stochastic inverse subspace iteration studied in [67, 86] for solution of eq. (4.1). The approach derives from a stochastic Galerkin formulation of subspace iteration, which is based on projection onto a finite-dimensional subspace of $L^2(\Gamma)$ spanned by the gPC basis functions $\{\psi_k(\xi)\}_{k=1}^{n_\xi}$. These functions are orthonormal, with

$$\langle \psi_i \psi_j \rangle = \mathbb{E}[\psi_i \psi_j] = \int_\Gamma \psi_i(\xi)\psi_j(\xi)\mathrm{d}\mu = \delta_{ij}, \tag{4.2}$$

where $\langle \cdot \rangle$ is the expected value, and δ_{ij} is the Kronecker delta. The stochastic Galerkin solutions are expressed as expansions of the gPC basis functions,

$$\lambda^s(\xi) = \sum_{r=1}^{n_\xi} \lambda_r^s \psi_r(\xi), \quad u^s(\xi) = \sum_{j=1}^{n_\xi} u_j^s \psi_j(\xi). \tag{4.3}$$

We briefly review the stochastic subspace iteration method in the case where $A(\xi)$ admits an affine expansion with respect to components of the random variable ξ:

$$A(\xi) = A_0 + \sum_{l=1}^{m} A_l \xi_l \tag{4.4}$$

where each A_l is an $n_x \times n_x$ deterministic matrix, obtained from, for instance, finite element discretization of a PDE operator. The matrix A_0 is the mean value of $A(\xi)$. Such a representation can be obtained from a KL expansion of the stochastic term

in the problem. Let $\{u^{s,(i)}(\xi)\}_{s=1}^{n_e}$ be a set of approximate eigenvectors obtained at the ith step of the inverse subspace iteration. Then at step $i+1$, one needs to solve

$$\langle Av^{s,(i+1)}\psi_k\rangle = \langle u^{s,(i)}\psi_k\rangle, \quad k = 1, 2, \ldots, n_\xi, \tag{4.5}$$

for $\{v^{s,(i+1)}\}_{s=1}^{n_e}$ and compute $\{u^{s,(i+1)}\}_{s=1}^{n_e}$ via orthonormalization. If $n_e = 1$, for the latter requirement, $v^{s,(i+1)}$ is normalized so that $||u^{s,(i+1)}||_2 = 1$ almost surely. If $n_e > 1$, a stochastic version of the Gram–Schmidt process is applied and the resulting vectors $\{u^{s,(i+1)}\}_{s=1}^{n_e}$ satisfy $\langle u^{s,(i+1)}, u^{t,(i+1)}\rangle_{\mathbb{R}^{n_x}} = \delta_{st}$ almost surely, where $\langle\cdot,\cdot\rangle_{\mathbb{R}^{n_x}}$ is the Euclidean inner product in $\mathbb{R}^{n_x}$. With the iterates expressed as gPC expansions, for instance, $u^{s,(i)}(\xi) = \sum_{j=1}^{n_\xi} u_j^{s,(i)}\psi_j(\xi)$, collecting the n_ξ equations in eq. (4.5) for each s yields an $n_x n_\xi \times n_x n_\xi$ linear system

$$\sum_{l=0}^{m}(G_l \otimes A_l)\mathbf{v}^{s,(i+1)} = \mathbf{u}^{s,(i)} \tag{4.6}$$

where $\otimes$ is the Kronecker product, each G_l is an $n_\xi \times n_\xi$ matrix with $[G_l]_{kj} = \langle \xi_l \psi_k \psi_j\rangle$ ($\xi_0 \equiv 1$ and $G_0 = I$), and

$$\mathbf{u}^{s,(i)} = \begin{pmatrix} u_1^{s,(i)} \\ u_2^{s,(i)} \\ \vdots \\ u_{n_\xi}^{s,(i)} \end{pmatrix} \in \mathbb{R}^{n_x n_\xi}. \tag{4.7}$$

Note that the matrices $\{G_l\}$ are sparse due to orthogonality of the gPC basis functions [29, 74]. The initial iterate is given by solving the mean problem $A_0\bar{u}^s = \bar{\lambda}^s\bar{u}^s$,

and

$$\mathbf{u}^{s,(0)} = \begin{pmatrix} \bar{u}^s \\ 0 \\ \vdots \\ 0 \end{pmatrix}. \tag{4.8}$$

The complete algorithm is summarized as algorithm 4.1. The details of the computations in steps 4 and 7 are given in eqs. (4.13), (4.19) and (4.22) below.

Algorithm 4.1: Stochastic inverse subspace iteration

1: **initialization**: initial iterate $\mathbf{u}^{s,(0)}$.
2: **for** $i = 0, 1, 2, \ldots$ **do**
3: Solve the stochastic Galerkin system eq. (4.6) for $\mathbf{v}^{s,(i+1)}$, $s = 1, 2, \ldots, n_e$.
4: If $n_e = 1$, compute $\mathbf{u}^{s,(i+1)}$ by normalization. Otherwise, apply a stochastic Gram–Schmidt process for orthonormalization.
5: Check convergence.
6: **end**
7: Compute eigenvalues using a Rayleigh quotient.

4.3 Low-rank approximation

In this section we discuss the idea of low-rank approximation and how this can be used to reduce the computational costs of algorithm 4.1. The size of the Galerkin system eq. (4.6) is in general large and solving the system can be computationally expensive. We utilize low-rank iterative solvers where the iterates are approximated by low-rank matrices and the system is efficiently solved to a specified accuracy. In addition, low-rank forms can be used to reduce the costs of the orthonormalization and Rayleigh quotient computations in the algorithm.

4.3.1 System solution

For any random vector $x(\xi)$ with expansion $x(\xi) = \sum_{j=1}^{n_\xi} x_j \psi_j(\xi)$ where each x_j is a vector of length n_x, let

$$X = \text{mat}(\mathbf{x}) = [x_1, x_2, \ldots, x_{n_\xi}] \in \mathbb{R}^{n_x \times n_\xi}. \tag{4.9}$$

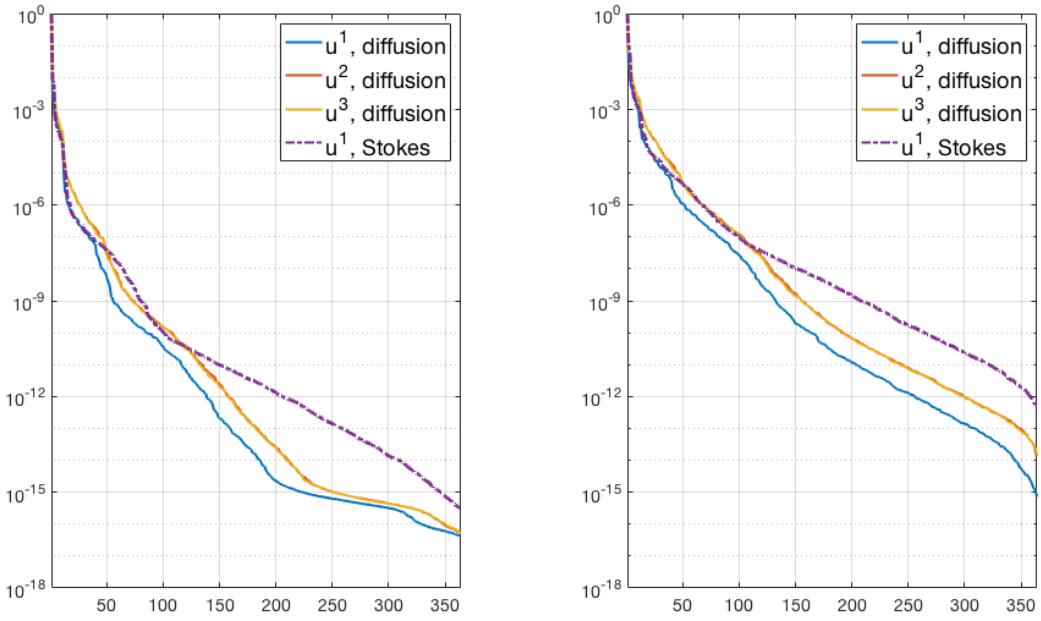
Then the Galerkin system $\sum_{l=0}^{m} (G_l \otimes A_l)\mathbf{x} = \mathbf{f}$ is equivalent to the matrix form

$$\sum_{l=0}^{m} A_l X G_l^T = F = \text{mat}(\mathbf{f}). \tag{4.10}$$

For such a matrix equation obtained from the stochastic Galerkin discretization, the approach in [7] can be applied to show that the solution X can be approximated by a low-rank matrix. In particular, in examples considered in this study, when the variance of the random parameters is small, the singular values of the solution matrix decay exponentially fast (see fig. 4.1), and a low-rank approximate solution can be obtained by dropping the terms corresponding to small singular values in a singular value decomposition.

To take advantage of the low rank of the solution matrix, we construct iterative solvers that produce a sequence of low-rank approximate iterates. Let $X^{(i)} = \text{mat}(\mathbf{x}^{(i)})$ be the ith iterate computed by an iterative solver applied to eq. (4.10), and suppose $X^{(i)}$ is represented as the product of two rank-κ matrices, i.e., $X^{(i)} = Y^{(i)} Z^{(i)T}$, where $Y^{(i)} \in \mathbb{R}^{n_x \times \kappa}$, $Z^{(i)} \in \mathbb{R}^{n_\xi \times \kappa}$. If this factored form is used throughout the iteration without explicitly forming $X^{(i)}$, then the matrix-vector product $(G_l \otimes A_l)\mathbf{x}$ will have the same structure,

$$A_l X^{(i)} G_l^T = (A_l Y^{(i)})(G_l Z^{(i)})^T, \tag{4.11}$$



(a) $\sigma = 0.01$ (b) $\sigma = 0.1$

Figure 4.1: Singular values (relative to the largest one) of the matrix representations of the stochastic eigenvectors for the numerical examples in sections 4.4 and 4.5, with standard deviations $\sigma = 0.01$ and $\sigma = 0.1$. $n_c = 5$, $b = 4.0$, $m = 11$, $n_\xi = 364$.

and it is only necessary to compute $A_l Y^{(i)}$ and $G_l Z^{(i)}$. If $\kappa \ll \min(n_x, n_\xi)$, this means that the computational costs of the matrix operation are reduced from $O(n_x n_\xi)$ to $O((n_x + n_\xi)\kappa)$. On the other hand, summing terms with the factored form tends to increase the rank, and rank compression techniques must be used in each iteration to force the matrix rank κ to stay low. In particular, if $X_1^{(i)} = Y_1^{(i)} Z_1^{(i)T}$, $X_2^{(i)} = Y_2^{(i)} Z_2^{(i)T}$, where $Y_1^{(i)} \in \mathbb{R}^{n_x \times \kappa_1}$, $Z_1^{(i)} \in \mathbb{R}^{n_\xi \times \kappa_1}$, $Y_2^{(i)} \in \mathbb{R}^{n_x \times \kappa_2}$, $Y_2^{(i)} \in \mathbb{R}^{n_\xi \times \kappa_2}$, then

$$X_1^{(i)} + X_2^{(i)} = [Y_1^{(i)}, Y_2^{(i)}][Z_1^{(i)}, Z_2^{(i)}]^T. \tag{4.12}$$

The addition gives a matrix of rank $\kappa_1 + \kappa_2$ in the worst case. This issue is the same as that discussed for the multigrid solver in chapter 3. Rank compression can be achieved by an SVD-based truncation operator $\tilde{X}^{(i)} = \mathcal{T}(X^{(i)})$ so the matrix $\tilde{X}^{(i)}$ has a much smaller rank than $X^{(i)}$ [56] (see also section 3.3 for a detailed discussion of the truncation operator). Low-rank approximation and truncation have been used for Krylov subspace methods [7, 56, 60] and multigrid methods (chapter 3, [27]). More details can be found in these references. We will use examples of such solvers for linear systems arising in eigenvalue computations, as discussed in sections 4.4 and 4.5.

4.3.2 Orthonormalization

In algorithm 4.1, if $n_e = 1$, the solution $v^{s,(i+1)}(\xi)$ is normalized so that $||u^{s,(i+1)}(\xi)||_2 = 1$ almost surely. With the superscripts omitted, assume $u(\xi) = \sum_{j=1}^{n_\xi} u_j \psi_j(\xi)$ is the normalized random vector constructed from $v(\xi)$. This expan-

sion can be computed using sparse grid quadrature $\{\xi^{(q)}, \eta^{(q)}\}_{q=1}^{n_q}$, where $\{\eta^{(q)}\}$ are the weights [32]:

$$u_j = \langle u(\xi)\psi_j(\xi)\rangle = \left\langle \frac{v(\xi)}{\|v(\xi)\|_2}\psi_j(\xi) \right\rangle \approx \sum_{q=1}^{n_q} \frac{v(\xi^{(q)})}{\|v(\xi^{(q)})\|_2}\psi_j(\xi^{(q)})\eta^{(q)}. \tag{4.13}$$

Suppose the "matricized" version of the expansion coefficients of $v(\xi)$ is represented in low-rank form

$$V = [v_1, v_2, \ldots, v_{n_\xi}] = Y_v Z_v^T, \tag{4.14}$$

where $Y_v \in \mathbb{R}^{n_x \times \kappa_v}$, $Z_v \in \mathbb{R}^{n_\xi \times \kappa_v}$. With $\Psi(\xi^{(q)}) = [\psi_1(\xi^{(q)}), \psi_2(\xi^{(q)}), \ldots, \psi_{n_\xi}(\xi^{(q)})]^T$, we have

$$v(\xi^{(q)}) = \sum_{j=1}^{n_\xi} v_j \psi_j(\xi^{(q)}) = V\Psi(\xi^{(q)}) = Y_v Z_v^T \Psi(\xi^{(q)}). \tag{4.15}$$

Let $U = [u_1, u_2, \ldots, u_{n_\xi}]$. Then eq. (4.13) yields

$$[U]_{:,j} = u_j = \sum_{q=1}^{n_q} \frac{Y_v Z_v^T \Psi(\xi^{(q)})}{\|Y_v Z_v^T \Psi(\xi^{(q)})\|_2}\psi_j(\xi^{(q)})\eta^{(q)}, \tag{4.16}$$

and

$$U = \sum_{q=1}^{n_q} \frac{Y_v Z_v^T \Psi(\xi^{(q)})}{\|Y_v Z_v^T \Psi(\xi^{(q)})\|_2}\Psi(\xi^{(q)})^T\eta^{(q)}. \tag{4.17}$$

Thus, the matrix U can be expressed as an outer product of two low-rank matrices $U = Y_u Z_u^T$ with

$$Y_u = Y_v \in \mathbb{R}^{n_x \times \kappa_v}, \quad Z_u = \sum_{q=1}^{n_q} \frac{\Psi(\xi^{(q)})(\Psi(\xi^{(q)})^T Z_v)}{\|Y_v(Z_v^T\Psi(\xi^{(q)}))\|_2}\eta^{(q)} \in \mathbb{R}^{n_\xi \times \kappa_v}. \tag{4.18}$$

This implies that the expansion coefficients of the normalized vector $u(\xi)$ can be written as a low-rank matrix with the same rank as the analogous matrix associated with $v(\xi)$. The cost of computing Z_u is $O((n_x + n_\xi)n_q\kappa_v)$. Since in general $n_q \gg \kappa_v$, it can be further reduced to $O((n_x + n_q)\kappa_v^2 + n_\xi n_q \kappa_v)$ by first computing a QR factorization of Y_v and factoring out the orthogonal matrix in the denominator.

In the general case where more than one eigenvector is computed ($n_e > 1$), a stochastic version of the Gram–Schmidt process is applied to compute an orthonormal set $\{u^{s,(i+1)}(\xi)\}_{s=1}^{n_e}$ [67, 86]. With the superscript $(i+1)$ omitted, the process is based on the following calculation

$$u^s(\xi) = v^s(\xi) - \sum_{t=1}^{s-1} \chi^{ts}(\xi) = v^s(\xi) - \sum_{t=1}^{s-1} \frac{\langle v^s(\xi), u^t(\xi)\rangle_{\mathbb{R}^{n_x}}}{\langle u^t(\xi), u^t(\xi)\rangle_{\mathbb{R}^{n_x}}} u^t(\xi) \tag{4.19}$$

for $s = 2, \ldots, n_e$. If we write $\chi^{ts}(\xi) = \sum_{k=1}^{n_\xi} \chi_k^{ts} \psi_k(\xi)$, and assume $u^t(\xi)$ is already normalized in previous steps, then

$$\begin{aligned} \chi_k^{ts} &= \langle v^s(\xi)^T u^t(\xi) u^t(\xi) \psi_k(\xi)\rangle \\ &\approx \sum_{q=1}^{n_q} v^s(\xi^{(q)})^T u^t(\xi^{(q)}) u^t(\xi^{(q)}) \psi_k(\xi^{(q)}) \eta^{(q)} \\ &= \sum_{q=1}^{n_q} (\Psi(\xi^{(q)})^T Z_{v^s} Y_{v^s}^T)(Y_{u^t} Z_{u^t}^T \Psi(\xi^{(q)})) Y_{u^t} Z_{u^t}^T \Psi(\xi^{(q)}) \psi_k(\xi^{(q)}) \eta^{(q)}. \end{aligned} \tag{4.20}$$

The last line follows eq. (4.15). Let $\zeta^{ts}(\xi^{(q)}) = (\Psi(\xi^{(q)})^T Z_{v^s} Y_{v^s}^T)(Y_{u^t} Z_{u^t}^T \Psi(\xi^{(q)}))$, then the matrix $X^{ts} = [\chi_1^{ts}, \chi_2^{ts}, \ldots, \chi_{n_\xi}^{ts}]$ can be expressed in low-rank form $X^{ts} = Y_{\chi^{ts}} Z_{\chi^{ts}}^T$ with

$$Y_{\chi^{ts}} = Y_{u^t}, \quad Z_{\chi^{ts}} = \sum_{q=1}^{n_q} \Psi(\xi^{(q)}) (\Psi(\xi^{(q)})^T Z_{u^t}) \zeta^{ts}(\xi^{(q)}) \eta^{(q)}. \tag{4.21}$$

With low-rank representaion, the computational cost is $O((n_x + n_\xi) n_q \max(\kappa_{v^s}, \kappa_{u^t}))$. Note that in eq. (4.19) the summation will increase the matrix rank, and thus a truncation operator is applied to compress the rank.

4.3.3 Rayleigh quotient

The Rayleigh quotient in step 7 of algorithm 4.1 is computed (only once) after convergence of the inverse subspace iteration to find the eigenvalues. Given a

normalized eigenvector $u(\xi)$ of problem eq. (4.1), the computation of the stochastic Rayleigh quotient

$$\lambda(\xi) = u(\xi)^T A(\xi) u(\xi) \tag{4.22}$$

involves two steps:

(1) Compute matrix-vector product $w(\xi) = A(\xi)u(\xi)$ where $w(\xi) = \sum_{k=1}^{n_\xi} w_k \psi_k(\xi)$ and $w_k = \langle Au\psi_k \rangle$. In Kronecker product form,

$$\mathbf{w} = \sum_{l=0}^{m} (G_l \otimes A_l)\mathbf{u}. \tag{4.23}$$

If $\mathbf{u}$ has low-rank representation $U = Y_u Z_u^T$, then

$$W = \sum_{l=0}^{m} (A_l Y_u)(G_l Z_u)^T. \tag{4.24}$$

This is followed by a truncation operation to compress the matrix rank.

(2) Compute eigenvalue $\lambda(\xi) = u(\xi)^T w(\xi)$ where $\lambda(\xi) = \sum_{r=1}^{n_\xi} \lambda_r \psi_r(\xi)$ and $\lambda_r = \langle u^T w \psi_r \rangle$. Equivalently,

$$\lambda_r = \langle H_r, C \rangle_{\mathbb{R}^{n_\xi \times n_\xi}} = \sum_{j,k=1}^{n_\xi} [H_r]_{jk} C_{jk} \tag{4.25}$$

where $C_{jk} = u_j^T w_k$ and thus $C = U^T W = Z_u (Y_u^T Y_w) Z_w^T$. The matrices $\{H_r\}_{r=1}^{n_\xi}$ are sparse with $[H_r]_{jk} = \langle \psi_r \psi_j \psi_k \rangle$. In fact, if the basis functions are written as products of univariate polynomials, i.e.,

$$\psi_r(\xi) = \psi_{r_1}(\xi_1)\psi_{r_2}(\xi_2)\cdots\psi_{r_m}(\xi_m), \tag{4.26}$$

then $[H_r]_{jk}$ is nonzero only if $|j_l - k_l| \le r_l \le j_l + k_l$ and $r_l + j_l + k_l$ is even for all $1 \le l \le m$ [29]. This observation greatly reduces the cost of assembling the

matrices $\{H_r\}$. For example, if $m = 11$, the degree of the gPC basis functions is $p \leq 3$, and $n_\xi = (m+p)!/(m!p!) = 364$, then with the above rule, a total of 31098 nonzero entries must be computed, instead of the much larger number $n_\xi^3 = 48228544$ if the sparsity of $\{H_r\}$ is not used.

4.3.4 Convergence criterion

To check convergence, we can look at the magnitude of the expected value of the residual

$$r^s(\xi) = A(\xi)u^s(\xi) - \lambda^s(\xi)u^s(\xi), \quad s = 1, 2, \ldots, n_e. \tag{4.27}$$

Alternatively, without computing the Rayleigh quotient at each iteration, error assessment can be done using the relative difference of the gPC coefficients of two successive iterates, i.e.,

$$\epsilon_{\Delta u}^{s,(i)} = \frac{1}{n_\xi} \sum_{k=1}^{n_\xi} \frac{\|u_k^{s,(i)} - u_k^{s,(i-1)}\|_2}{\|u_k^{s,(i-1)}\|_2}. \tag{4.28}$$

However, in the case of clustered eigenvalues (that is, if two or more eigenvalues are close to each other), the convergence of the inverse subspace iteration for single eigenvectors will be slow. Instead, we look at the angle between the eigenspaces [12] in two consecutive iterations

$$\theta^{(i)}(\xi) = \angle(\text{span}(u^{1,(i)}(\xi), \ldots, u^{n_e,(i)}(\xi)), \text{span}(u^{1,(i-1)}(\xi), \ldots, u^{n_e,(i-1)}(\xi))). \tag{4.29}$$

The expected value $\mathbb{E}[\theta^{(i)}]$ is taken as error indicator and is also calculated using sparse grid quadrature

$$\epsilon_\theta^{(i)} = \mathbb{E}[\theta^{(i)}] \approx \sum_{q=1}^{n_q} \theta^{(i)}(\xi^{(q)})\eta^{(q)}. \tag{4.30}$$

At each quadrature point, $\theta^{(i)}(\xi^{(q)})$ is evaluated by MATLAB function `subspace` for the largest principal angle.

4.4 Stochastic diffusion equation

In this section we consider the following elliptic equation with Dirichlet boundary conditions

$$\begin{cases} -\nabla \cdot (a(x,\omega)\nabla u(x,\omega)) = \lambda(\omega)u(x,\omega) & \text{in } \mathcal{D}\times\Omega \\ u(x,\omega) = 0 & \text{on } \partial\mathcal{D}\times\Omega \end{cases} \tag{4.31}$$

where $\mathcal{D}$ is a two-dimensional spatial domain and Ω is a sample space. The uncertainty in the problem is introduced by the stochastic diffusion coefficient $a(x,\omega)$. Assume that $a(x,\omega)$ is bounded and strictly positive and admits a truncated KL expansion

$$a(x,\omega) = a_0(x) + \sum_{l=1}^{m} \sqrt{\beta_l} a_l(x)\xi_l(\omega), \tag{4.32}$$

where $a_0(x)$ is the mean function, $(\beta_l, a_l(x))$ is the lth eigenpair of the covariance function, and $\{\xi_l\}$ are a collection of uncorrelated random variables. The weak form is to find $(u(x,\xi), \lambda(\xi))$ such that for any $v(x) \in H_0^1(\mathcal{D})$,

$$\int_{\mathcal{D}} a(x,\xi)\nabla u(x,\xi)\cdot\nabla v(x)\mathrm{d}x = \lambda(\xi)\int_{\mathcal{D}} u(x,\xi)v(x)\mathrm{d}x \tag{4.33}$$

almost surely.

Finite element discretization in the physical domain $\mathcal{D}$ with basis functions $\{\phi_i(x)\}$ gives

$$K(\xi)u(\xi) = \lambda(\xi)Mu(\xi) \tag{4.34}$$

where $K(\xi) = \sum_{l=0}^{m} K_l \xi_l$, and

$$\begin{aligned} [K_l]_{ij} &= \int_{\mathcal{D}} \sqrt{\beta_l} a_l(x) \nabla \phi_i(x) \cdot \nabla \phi_j(x) \mathrm{d}x, \\ [M]_{ij} &= \int_{\mathcal{D}} \phi_i(x) \phi_j(x) \mathrm{d}x, \quad i, j = 1, 2, \ldots, n_x, \end{aligned} \tag{4.35}$$

with $\beta_0 = 1$ and $\xi_0 \equiv 1$. The result is a generalized eigenvalue problem where the matrix M on the right-hand side is deterministic. With the Cholesky factorization $M = LL^T$, eq. (4.34) can be converted to standard form

$$A(\xi) w(\xi) = \lambda(\xi) w(\xi), \tag{4.36}$$

where $A(\xi) = L^{-1} K(\xi) L^{-T}$, $w(\xi) = L^T u(\xi)$.

We use stochastic inverse subspace iteration to find n_e minimal eigenvalues of eq. (4.36). As discussed in section 4.2, the linear systems to be solved in each iteration are in the form

$$\sum_{l=0}^{m} (G_l \otimes (L^{-1} K_l L^{-T})) \mathbf{v}^{s,(i+1)} = \mathbf{u}^{s,(i)}, \quad s = 1, 2, \ldots, n_e. \tag{4.37}$$

Let $\mathbf{v}^{s,(i)} = (I \otimes L^T) \hat{\mathbf{v}}^{s,(i)}$. Then eq. (4.37) is equivalent to

$$\sum_{l=0}^{m} (G_l \otimes K_l) \hat{\mathbf{v}}^{s,(i+1)} = (I \otimes L) \mathbf{u}^{s,(i)}. \tag{4.38}$$

4.4.1 Low-rank multigrid

We discussed a low-rank geometric multigrid method in chapter 3 for solving linear systems with the same structure as eq. (4.38). The complete algorithm for solving $\mathscr{A}(X) = F$ is given in algorithm 3.2, where $\mathscr{A}$ is a generic matrix operator and for eq. (4.38), $\mathscr{A}(X) = \sum_{l=0}^{m} K_l X G_l^T$. All the iterates are expressed in low-rank form, and truncation operations are used to compress the ranks of the iterates.

$\mathcal{T}_{\text{rel}}$ and $\mathcal{T}_{\text{abs}}$ are truncation operators with a relative tolerance ϵ_{rel} and an absolute tolerance ϵ_{abs}, respectively. In each iteration, one V-cycle is applied to the residual equation. On the coarse grids, coarse versions of $\{K_l\}$ are assembled while the matrices $\{G_l\}$ stay the same. The prolongation operator is $\mathscr{P} = I \otimes P$, where P is the same prolongation matrix as in a standard geometric multigrid solver, and the restriction operator is $\mathscr{R} = I \otimes P^T$. The smoothing operator $\mathscr{S}$ is based on a stationary iteration, and is also a Kronecker product of two matrices. The grid transfer and smoothing operations do not affect the rank. For instance, for any matrix iterate in low-rank form $X^{(i)} = Y^{(i)} Z^{(i)T}$,

$$\mathscr{P}(X^{(i)}) = (PY^{(i)})(IZ^{(i)})^T. \tag{4.39}$$

On the coarsest grid ($h = h_0$), the system is solved with direct methods.

4.4.2 Rayleigh–Ritz refinement

It is known that in the deterministic case with a constant diffusion coefficient, eq. (4.34) typically has repeated eigenvalues [26], for example, $\lambda^2 = \lambda^3$. The parametrized versions of these eigenvalues in the stochastic problem will be close to each other. In the deterministic setting, Rayleigh–Ritz refinement is used to accelerate the convergence of subspace iteration when some eigenvalues have nearly equal modulus and the convergence to individual eigenvectors is slow [88, 89]. Assume that a Hermitian matrix S has eigendecomposition

$$S = V\Lambda V^T = V_1\Lambda_1 V_1^T + V_2\Lambda_2 V_2^T \tag{4.40}$$

where $\Lambda = \mathrm{diag}(\lambda^1, \lambda^2, \ldots, \lambda^{n_x})$ with eigenvalues in increasing order and $V = [V_1, V_2]$ is orthogonal. Let the column space of Q be a good approximation to that of V_1. Such an approximation is obtained from the inverse subspace iteration. The Rayleigh–Ritz procedure computes

(1) Rayleigh quotient $T = Q^T S Q$, and

(2) eigendecomposition $T = W \Sigma W^T$.

Then Σ and QW represent good approximations to Λ_1 and V_1.

The stochastic inverse subspace iteration algorithm produces solutions $\{u^s_{\mathrm{SG}}(\xi)\}$ expressed as gPC expansions as in eq. (4.3) and sample eigenvectors are easily computed. The sample eigenvalues are generated from the stochastic Rayleigh quotient eq. (4.22). However, in the case of poorly separated eigenvalues, the sample solutions obtained this way are not accurate enough. Experimental results that demonstrate this are given in section 4.4.3, see table 4.2. Instead, we use a version of the Rayleigh–Ritz procedure to generate sample eigenvalues and eigenvectors with more accuracy. Specifically, a parametrized Rayleigh quotient $T(\xi)$ is computed using the approach of section 4.3.3, with

$$[T]_{st}(\xi) = u^s_{\mathrm{SG}}(\xi)^T A(\xi) u^t_{\mathrm{SG}}(\xi), \quad s, t = 1, 2, \ldots, n_e. \tag{4.41}$$

Then one can sample the matrix T, and for each realization $\xi^{(r)}$, solve a small $(n_e \times n_e)$ deterministic eigenvalue problem $T(\xi^{(r)}) = W(\xi^{(r)}) \Sigma(\xi^{(r)}) W(\xi^{(r)})^T$ to get

better approximations for the minimal eigenvalues and corresponding eigenvectors:

$$\begin{aligned}\tilde{\lambda}^s_{\mathrm{SG}}(\xi^{(r)}) &= [\Sigma(\xi^{(r)})]_{ss},\\ \tilde{u}^s_{\mathrm{SG}}(\xi^{(r)}) &= [u^1_{\mathrm{SG}}(\xi^{(r)}), u^2_{\mathrm{SG}}(\xi^{(r)}), \ldots, u^{n_e}_{\mathrm{SG}}(\xi^{(r)})][W(\xi^{(r)})]_{:,s}.\end{aligned} \tag{4.42}$$

The effectiveness of this procedure will also be demonstrated in section 4.4.3, see table 4.3.

4.4.3 Numerical experiments

Consider a two-dimensional domain $\mathcal{D} = [-1, 1]^2$. Let the spatial discretization consist of piecewise bilinear basis functions on a uniform square mesh. The finite element matrices are assembled using the IFISS software package [82]. The number of spatial degrees of freedom is $n_x = (2/h - 1)^2$ where h is the mesh size. Define the grid level n_c such that $2/h = 2^{n_c}$. In the KL expansion eq. (4.32), we use an exponential covariance function

$$c(x, y) = \sigma^2 \exp\left(-\frac{1}{b}\|x - y\|_1\right) \tag{4.43}$$

and $(\beta_l, a_l(x))$ is the lth eigenpair of $c(x, y)$. The correlation length b affects the decay of the eigenvalues $\{\beta_l\}$. The number of random variables m is chosen so that $(\sum_{l=1}^{m} \beta_l)/(\sum_{l=1}^{\infty} \beta_l) \geq 95\%$. Take the standard deviation $\sigma = 0.01$, the mean function $a_0(x) \equiv 1.0$, and $\{\xi_l\}$ to be independent and uniformly distributed on $[-\sqrt{3}, \sqrt{3}]^m$. Legendre polynomials are used for gPC basis functions, whose total degree does not exceed $p = 3$. The number of gPC basis functions is $n_\xi = (m + p)!/(m!p!)$. For the quadrature rule in section 4.3.2, we use a Smolyak sparse grid with Clenshaw–Curtis quadrature points and grid level 3, computed from the

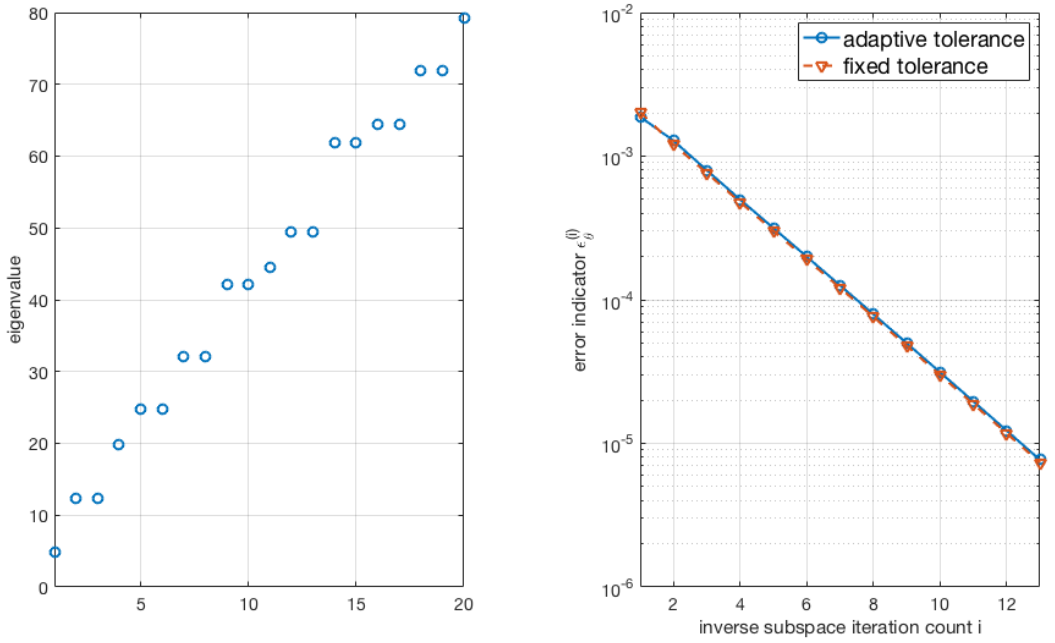
SPINTERP toolbox [52]. For $m = 11$, the number of sparse grid points is 2069. All computations in this chapter are done in MATLAB 9.4.0 (R2018a) on a MacBook with 4 GB SDRAM.

We apply low-rank stochastic inverse subspace iteration to compute three minimal eigenvalues ($n_e = 3$) and corresponding eigenvectors for eq. (4.34). The smallest 20 eigenvalues for the mean problem $K_0 u = \lambda M u$ are plotted in fig. 4.2a. For the stochastic problem, the three smallest eigenvalues consist of one isolated smallest eigenvalue $\lambda^1(\xi)$ and (as mentioned in the previous subsection) two eigenvalues $\lambda^2(\xi)$ and $\lambda^3(\xi)$ that have nearly equal modulus. For the inverse subspace iteration, we take $\epsilon_\theta^{(i)}$ in eq. (4.30) as error indicator and use a stopping criterion $\epsilon_\theta^{(i)} \leq tol_{\text{isi}} = 10^{-5}$. The low-rank multigrid method of section 4.4.1 is used to solve the system eq. (4.38), where damped Jacobi iteration is employed for the smoothing opeator $\mathscr{S} = \omega_s \text{diag}(\mathscr{A})^{-1} = \omega_s (I \otimes K_0^{-1})$ with weight $\omega_s = 2/3$. Two smoothing steps are applied ($\nu = 2$). We also use the idea of inexact inverse iteration methods [36, 58] so that in the first few steps of subspace iteration, the systems eq. (4.6) are solved with milder error tolerances than in later steps. Specifically, we set the multigrid tolerance as

$$tol_{\text{mg}}^{(i)} = \max\{\min\{10^{-2} * \epsilon_\theta^{(i-1)}, 10^{-3}\}, 10^{-6}\}, \tag{4.44}$$

and truncation tolerances $\epsilon_{\text{abs}}^{(i)} = 10^{-2} * tol_{\text{mg}}^{(i)}$, $\epsilon_{\text{rel}} = 10^{-2}$ [27]. This is shown to be useful in reducing the computational costs while not affecting the convergence of the subspace iteration algorithm (see fig. 4.2b).

Table 4.1 shows the ranks of the multigrid solutions in each iteration. It



(a) (b)

Figure 4.2: (a) Smallest 20 eigenvalues of the mean problem. (b) Reduction of the error indicator $\epsilon_\theta^{(i)}$ for an adaptive multigrid tolerance eq. (4.44) and a fixed tolerance $tol_{\text{mg}} = 10^{-6}$. $n_c = 6$, $b = 4.0$, $m = 11$.

indicates that all the systems solved have low-rank approximate solutions ($n_x = 3969$, $n_\xi = 364$). With the inexact solve, the solutions have much smaller ranks in the first few iterations. In the last row of table 4.1 are the numbers of multigrid steps it_{mg} required to solve eq. (4.38) for $s = 1$; similar numbers of multigrid steps are required for $s = 2, 3$. In addition, in algorithm 4.1 an absolute truncation operator with $\epsilon_{\text{abs}} = 10^{-8}$ is applied after the computations in eqs. (4.19) and (4.23) (both require addition of quantities represented as low-rank matrices in implementation) to compress the iterate ranks. Rayleigh–Ritz refinement discussed in section 4.4.2 is used to obtain good approximations to individual sample eigenpairs.

Table 4.1: Iterate ranks after the multigrid solve and numbers of multigrid steps required in the inverse subspace iteration algorithm. $n_c = 6$, $b = 4.0$, $m = 11$.

(i)		1	2	3	4	5	6	7	8	9	10	11	12	13
Rank	u^1	11	22	26	32	40	44	44	46	49	49	49	49	49
	u^2	17	23	25	33	41	41	41	41	41	41	41	41	41
	u^3	17	25	28	37	39	40	40	40	40	40	40	40	40
it_{mg}		3	5	5	6	6	6	6	7	7	7	7	7	7

To show the accuracy of the low-rank stochastic Galerkin solutions, we compare them with reference solutions from Monte Carlo simulation. The stochastic Galerkin method produces a surrogate stochastic solution expressed with gPC basis functions that can be easily sampled. The Monte Carlo solutions are computed by the `eigs` function from MATLAB, which uses the implicitly restarted Arnoldi method to compute several minimal eigenvalues [85]. For both methods, we use the same sample values $\{\xi^{(r)}\}$ of the random variables to generate sample eigenvalues

and eigenvectors. Define the relative errors

$$\epsilon_{\lambda^s} = \frac{1}{n_r}\sum_{r=1}^{n_r}\frac{|\lambda^s_{\mathrm{SG}}(\xi^{(r)}) - \lambda^s_{\mathrm{MC}}(\xi^{(r)})|}{|\lambda^s_{\mathrm{MC}}(\xi^{(r)})|},$$
$$\epsilon_{u^s} = \frac{1}{n_r}\sum_{r=1}^{n_r}\frac{\|u^s_{\mathrm{SG}}(\xi^{(r)}) - u^s_{\mathrm{MC}}(\xi^{(r)})\|_2}{\|u^s_{\mathrm{MC}}(\xi^{(r)})\|_2}, \tag{4.45}$$

where λ^s_{SG} and u^s_{SG} denote the stochastic Galerkin sample solutions (they are replaced by $\tilde{\lambda}^s_{\mathrm{SG}}$ and $\tilde{u}^s_{\mathrm{SG}}$ in eq. (4.42) if Rayleigh–Ritz refinement is used), λ^s_{MC} and u^s_{MC} are the Monte Carlo solutions, n_r is the sample size, and $s = 1, 2, \ldots, n_e$. We use a sample size $n_r = 10000$.

We examine the accuracy for the three smallest eigenvalues obtained from inverse subspace iteration when they are computed both with and without Rayleigh–Ritz refinement. table 4.2 shows the results (for one spatial mesh size) when Rayleigh–Ritz refinement is not used. It can be seen that (the poorly separated) eigenvalues λ^2 and λ^3 are significantly less accurate than λ^1, and that the eigenvectors u^2 and u^3 are highly inaccurate. In contrast, table 4.3 (with results for three mesh sizes) demonstrates dramatically improved accuracy when refinement is done. In all cases, convergence takes 13 iterations.

Table 4.2: Relative differences between low-rank stochastic Galerkin solutions (without Rayleigh–Ritz refinement) and Monte Carlo solutions. $n_c = 6$, $b = 4.0$, $m = 11$.

ϵ_{λ^1}	4.8752×10^{-10}	ϵ_{u^1}	2.2318×10^{-7}
ϵ_{λ^2}	5.1938×10^{-4}	ϵ_{u^2}	5.2216×10^{-1}
ϵ_{λ^3}	5.1872×10^{-4}	ϵ_{u^3}	5.2215×10^{-1}

There are several things to consider in order to assess the efficiency of the low-rank algorithm. First, note that the stochastic Galerkin method depends on

Table 4.3: Relative differences between low-rank stochastic Galerkin solutions (with Rayleigh–Ritz refinement) and Monte Carlo solutions. $b = 4.0$, $m = 11$.

n_c	6	7	8
ϵ_{λ^1}	4.8753×10^{-10}	4.8789×10^{-10}	4.8777×10^{-10}
ϵ_{λ^2}	1.7339×10^{-9}	1.7996×10^{-9}	1.7856×10^{-9}
ϵ_{λ^3}	1.6481×10^{-9}	1.7122×10^{-9}	1.7189×10^{-9}
ϵ_{u^1}	1.1390×10^{-7}	1.8687×10^{-7}	3.8855×10^{-7}
ϵ_{u^2}	8.2047×10^{-6}	8.3449×10^{-6}	8.5969×10^{-6}
ϵ_{u^3}	8.2795×10^{-6}	8.4110×10^{-6}	8.6885×10^{-6}

two separate computations, the inverse subspace iteration algorithm to compute the surrogate stochastic solution, and the repeated evaluation of the surrogate solution, to be done in a simulation. (The associated costs are denoted as t_{solve} and t_{sample} respectively.) In the parlance of reduced basis methods [94], the first part can be viewed as an offline computation and the second part as an online computation. One issue is how the costs of each of these steps for the low-rank algorithm compare with a more standard version of inverse subspace iteration that does not use low-rank constructions, which we refer to as the full-rank version. In contrast, each step of the Monte Carlo method requires the solution of a single eigenvalue problem. The cost of this computation will be much smaller than that of the offline computation required for the stochastic Galerkin method, but each step of a Monte Carlo simulation will be more costly than when a surrogate approximation is used.

Thus, the efficiency of the low-rank algorithm is demonstrated by comparison with (i) stochastic inverse subspace iteration with the full-rank stochastic Galerkin

method, with the same tolerances tol_{isi} and tol_{mg}, and (ii) the Monte Carlo method. For the latter method, each deterministic eigenvalue problem is now solved by an LOBPCG method [54], preconditioned with one V-cycle of AMG of the mean matrix K_0, using a stopping tolerance 10^{-3} for the norm of the eigenvalue residual $\|K(\xi^{(r)})u_{\text{MC}}(\xi^{(r)}) - \lambda_{\text{MC}}(\xi^{(r)})Mu_{\text{MC}}(\xi^{(r)})\|_2$, chosen so that LOBPCG produces sample solutions of accuracy comparable to that obtained using the stochastic Galerkin approach. (There are choices for the deterministic solver used for Monte Carlo. We also tried `eigs` with a mild stopping tolerance. For this (diffusion) problem, we found the costs of `eigs` and LOBPCG to be similar; however, LOBPCG is more efficient for the Stokes problem considered in section 4.5 below, since it does not require solving linear systems associated with $BK(\xi^{(r)})^{-1}B^T$ for each sample $\xi^{(r)}$. We used LOBPCG for all cost assessments.)

Computational costs are shown in table 4.4. It can be seen that the low-rank approximation greatly reduces both t_{solve} and t_{sample} for the stochastic Galerkin approach, especially as the mesh size is refined. Moreover, the total time required by the low-rank stochastic Galerkin method is much less than that for the Monte Carlo method with a sample size $n_r = 10000$, whereas the full-rank counterpart can be more expensive than Monte Carlo. This will be discussed further in section 4.5 below (see section 4.5.2).

More details on the computational costs of the low-rank stochastic Galerkin method are given in table 4.6 for various n_x and n_ξ. The table shows the percentages of t_{solve} used for the low-rank multigrid solver (t_{mg}), the Gram–Schmidt process (t_{gs}), the convergence criterion (t_{err}), and the Rayleigh quotient (t_{rq}) in the stochastic

Table 4.4: Time comparison (in seconds) between stochastic Galerkin method and Monte Carlo simulation for various n_c. $b = 4.0$, $m = 11$, $n_\xi = 364$, $n_r = 10000$.

n_c		6	7	8
n_x		3969	16129	65025
low-rank SG	t_{solve}	265.63	792.06	2971.15
	t_{sample}	4.16	15.41	66.17
full-rank SG	t_{solve}	452.11	1898.85	19699.92
	t_{sample}	25.29	94.70	426.99
MC		385.39	1989.60	8897.27

inverse subspace iteration algorithm. It is clear that the dominant cost is that associated with solving the linear systems. As n_ξ increases, the percentages of time for the Gram–Schmidt process and the Rayleigh quotient both increase, although they are still much smaller than that for system solves.

More details on the computational costs of the low-rank stochastic Galerkin method are given in table 4.6 for various n_x and n_ξ. The table shows the percentages of t_{solve} used for the low-rank multigrid solver (t_{mg}), the Gram–Schmidt process (t_{gs}), the convergence criterion (t_{err}), and the Rayleigh quotient (t_{rq}) in the stochastic inverse subspace iteration algorithm. It is clear that the dominant cost is that associated with solving the linear systems. The computation for checking convergence is also relatively expensive. As n_ξ increases, the percentages of time for the Gram-Schmidt process and the Rayleigh quotient both increase, although they are still much smaller than that for system solves.

Table 4.5: Time comparison (in seconds) between stochastic Galerkin method and Monte Carlo simulation for various m. $n_r = 10000$.

$m(b)$		8(5.0)	11(4.0)	16(3.0)
n_ξ		165	364	969
low-rank SG	t_{solve}	296.51	792.06	3198.15
	t_{sample}	11.56	15.41	22.56
full-rank SG	t_{solve}	642.16	1898.85	12229.23
	t_{sample}	45.77	94.70	260.40
MC		1963.53	1989.60	1809.25

(a) $n_c = 7, n_x = 16129$

$m(b)$		8(5.0)	11(4.0)	16(3.0)
n_ξ		165	364	969
low-rank SG	t_{solve}	1137.60	2971.15	10720.43
	t_{sample}	39.95	66.17	86.19
full-rank SG	t_{solve}	4673.44	19699.92	out of
	t_{sample}	194.66	426.99	memory
MC		7515.48	8897.27	8536.08

(b) $n_c = 8, n_x = 65025$

Table 4.6: Time consumption percentages for different parts of computations in the low-rank stochastic Galerkin method for various n_c and m.

$m(b)$		8(5.0)	11(4.0)	16(3.0)
n_ξ		165	364	969
$n_c = 7$ $n_x = 16129$	t_{mg}	79.84%	76.57%	72.46%
	t_{gs}	5.93%	8.11%	9.10%
	t_{err}	10.49%	11.68%	9.98%
	t_{rq}	1.62%	2.60%	8.02%
$n_c = 8$ $n_x = 65025$	t_{mg}	76.27%	74.54%	74.35%
	t_{gs}	6.50%	8.59%	8.84%
	t_{err}	11.30%	12.97%	12.05%
	t_{rq}	1.96%	2.05%	3.48%

4.5 Stochastic Stokes equation

The second example of a stochastic eigenvalue problem that we consider is used to estimate the inf-sup stability constant associated with a discrete stochastic Stokes problem. Consider the following stochastic incompressible Stokes equation in a two-dimensional domain

$$\begin{cases} -\nabla \cdot (a(x,\omega)\nabla \vec{u}(x,\omega)) + \nabla p(x,\omega) = \vec{0} & \text{in } \mathcal{D} \times \Omega \\ \nabla \cdot \vec{u}(x,\omega) = 0 & \text{in } \mathcal{D} \times \Omega \end{cases} \tag{4.46}$$

with a Dirichlet inflow boundary condition $\vec{u}(x,\omega) = \vec{u}_{\mathrm{D}}(x)$ on $\partial\mathcal{D}_{\mathrm{D}} \times \Omega$ and a Neumann outflow boundary condition $a(x,\omega)\nabla\vec{u}(x,\omega)\cdot\vec{n} - p(x,\omega)\vec{n} = \vec{0}$ on $\partial\mathcal{D}_{\mathrm{N}} \times \Omega$. Such problems and more general stochastic Navier–Stokes equations have been studied in [75, 87]. As in the diffusion problem, we assume that the stochastic viscosity $a(x,\omega)$ is represented by a truncated KL expansion eq. (4.32) with random variables $\{\xi_l\}_{l=1}^m$. The weak formulation of the problem is: find $\vec{u}(x,\xi)$ and $p(x,\xi)$ satisfying

$$\begin{cases} \int_{\mathcal{D}} a(x,\xi)\nabla\vec{u}(x,\xi) : \nabla\vec{v}(x) - p(x,\xi)\nabla\cdot\vec{v}(x)\,\mathrm{d}x = 0 \\ \int_{\mathcal{D}} q(x)\nabla\cdot\vec{u}(x,\xi)\,\mathrm{d}x = 0 \end{cases} \tag{4.47}$$

almost surely for any $\vec{v}(x) \in H_0^1(\mathcal{D})^2$ (zero boundary conditions on $\partial\mathcal{D}_{\mathrm{D}}$) and $q(x) \in L^2(\mathcal{D})$. Here $\nabla\vec{u} : \nabla\vec{v}$ is a componentwise scalar product ($\nabla u_{x_1}\cdot\nabla v_{x_1} + \nabla u_{x_2}\cdot\nabla v_{x_2}$ for two-dimensional (u_{x_1}, u_{x_2})). Finite element discretization with basis functions $\{\vec{\phi}_i(x)\}$ for the velocity field and $\{\varphi_k(x)\}$ for the pressure field results in a linear

system in the form

$$\begin{pmatrix} K(\xi) & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \vec{u}(\xi) \\ p(\xi) \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix}, \tag{4.48}$$

where $K(\xi) = \sum_{l=0}^{m} K_l \xi_l$, and

$$\begin{aligned} [K_l]_{ij} &= \int_{\mathcal{D}} \sqrt{\beta_l} a_l(x) \nabla \vec{\phi}_i(x) : \nabla \vec{\phi}_j(x) \mathrm{d}x, \\ [B]_{kj} &= -\int_{\mathcal{D}} \varphi_k(x) \nabla \cdot \vec{\phi}_j(x) \mathrm{d}x, \end{aligned} \tag{4.49}$$

for $i, j = 1, 2, \ldots, n_u$ and $k = 1, 2, \ldots, n_p$. The Dirichlet boundary condition is incorporated in the right-hand side.

We are interested in the parametrized inf-sup stability constant $\gamma(\xi)$ for the discrete problem. Evaluation of the inf-sup constant for various parameter values plays an important role for *a posteriori* error estimation for reduced basis methods [69, 94]. For this, we exploit the fact that $\gamma(\xi)$ has an algebraic interpretation [26]

$$\gamma^2(\xi) = \min_{q(\xi) \neq 0} \frac{\langle BK(\xi)^{-1} B^T q(\xi), q(\xi) \rangle_{\mathbb{R}^{n_p}}}{\langle Mq(\xi), q(\xi) \rangle_{\mathbb{R}^{n_p}}} \tag{4.50}$$

where M is the mass matrix with $[M]_{ij} = \int_{\mathcal{D}} \varphi_i(x) \varphi_j(x)$, $i, j = 1, 2, \ldots, n_p$. Thus, finding $\gamma(\xi)$ is equivalent to finding the smallest eigenvalue of the generalized eigenvalue problem

$$BK(\xi)^{-1} B^T q(\xi) = \lambda(\xi) M q(\xi) \tag{4.51}$$

associated with the stochastic pressure Schur complement $BK(\xi)^{-1}B^T$. This can be written in standard form as

$$L^{-1} BK(\xi)^{-1} B^T L^{-T} w(\xi) = \lambda(\xi) w(\xi) \tag{4.52}$$

where $M = LL^T$ is a Cholesky factorization, and $w(\xi) = L^T q(\xi)$.

The eigenvalue problem eq. (4.52) does not have exactly the same form as eq. (4.1), since it involves the inverse of $K(\xi)$. If we use the stochastic inverse iteration algorithm to compute the minimal eigenvalue of eq. (4.52), then each iteration requires solving

$$\langle L^{-1}BK^{-1}B^TL^{-T}v^{(i+1)}\psi_k\rangle = \langle u^{(i)}\psi_k\rangle, \quad k = 1, 2, \ldots, n_\xi, \tag{4.53}$$

for $v^{(i+1)}(\xi)$. We can reformulate eq. (4.53) to take advantage of the Kronecker product structure and low-rank solvers. Let $s(\xi) = -K(\xi)^{-1}B^TL^{-T}v^{(i+1)}(\xi)$ and let $\hat{v}^{(i+1)}(\xi) = L^{-T}v^{(i+1)}(\xi)$. Then eq. (4.53) is equivalent to the coupled system

$$\langle (Ks + B^T\hat{v}^{(i+1)})\psi_k\rangle = 0, \quad \langle Bs\psi_k\rangle = \langle -Lu^{(i)}\psi_k\rangle, \quad k = 1, 2, \ldots, n_\xi. \tag{4.54}$$

As discussed in section 4.2, the random vectors are expressed as gPC expansions. Thus, eq. (4.54) can be written in Kronecker product form as a discrete Stokes system for coefficient vectors $\mathbf{s}$, $\hat{\mathbf{v}}^{(i+1)}$,

$$\begin{pmatrix} \sum_{l=0}^m (G_l \otimes K_l) & I \otimes B^T \\ I \otimes B & 0 \end{pmatrix} \begin{pmatrix} \mathbf{s} \\ \hat{\mathbf{v}}^{(i+1)} \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ -(I \otimes L)\mathbf{u}^{(i)} \end{pmatrix}, \tag{4.55}$$

and $\mathbf{v}^{(i+1)} = (I \otimes L^T)\hat{\mathbf{v}}^{(i+1)}$.

In addition, for the eigenvalue problem eq. (4.52), computing the Rayleigh quotient eq. (4.22) requires solving a linear system. In the first step of eq. (4.22), for the matrix-vector product, one needs to compute $w(\xi) = K(\xi)^{-1}\hat{u}(\xi)$, where $\hat{u}(\xi) = B^TL^{-T}u(\xi)$. For the weak formulation, this corresponds to solving a linear system

$$\left(\sum_{l=0}^m G_l \otimes K_l\right)\mathbf{w} = \hat{\mathbf{u}}. \tag{4.56}$$

4.5.1 Low-rank MINRES

We discuss a low-rank iterative solver for eq. (4.55). The system is symmetric but indefinite, with a positive-definite $(1,1)$ block. A low-rank preconditioned MINRES method for solving $\mathscr{A}(X) = F$ is used and described in algorithm 4.2. The preconditioner is block-diagonal,

$$\mathscr{M} = \begin{pmatrix} \mathscr{M}_{11} & 0 \\ 0 & \mathscr{M}_{22} \end{pmatrix}. \tag{4.57}$$

We use an approximate mean-based preconditioner [74] for the $(1,1)$ block: $\mathscr{M}_{11} = G_0 \otimes \hat{K}_0 = I \otimes \hat{K}_0$. Here, $\hat{K}_0^{-1}$ is defined by approximation of the action of K_0^{-1}, using one V-cycle of AMG. For the $(2,2)$ block, we take $\mathscr{M}_{22} = I \otimes \hat{M}$, where the action of M^{-1} is approximated by 10 steps of Chebyshev iteration [83]. As in the multigrid method, all the quantities are in low-rank format, and truncation operations are applied to compress matrix ranks. algorithm 4.2 requires the computation of inner products of two low-rank matrices $\langle X_1, X_2 \rangle_{\mathbb{R}^{n_x \times n_\xi}}$. Let $X_1 = Y_1 Z_1^T$, $X_2 = Y_2 Z_2^T$ with $Y_1 \in \mathbb{R}^{n_x \times \kappa_1}$, $Z_1 \in \mathbb{R}^{n_\xi \times \kappa_1}$, $Y_2 \in \mathbb{R}^{n_x \times \kappa_2}$, $Z_2 \in \mathbb{R}^{n_\xi \times \kappa_2}$. Then the inner product can be computed with a cost of $O((n_x + n_\xi + 1)\kappa_1\kappa_2)$ [56]:

$$\langle X_1, X_2 \rangle = \mathrm{trace}(X_1^T X_2) = \mathrm{trace}(Z_1 Y_1^T Y_2 Z_2^T) = \mathrm{trace}((Z_2^T Z_1)(Y_1^T Y_2)). \tag{4.58}$$

We apply the low-rank MINRES method to the matricized version of eq. (4.55), and represent the components of the solution vector, $\mathbf{s}$ and $\hat{\mathbf{v}}^{(i+1)}$, as two separate low-rank matrices S and $\hat{V}^{(i+1)}$. This representation is suitable for computing matrix-vector products. For instance, the first equation becomes $\sum_{l=1}^{m} K_l S G_l^T +$

$B^T \hat{V}^{(i+1)} I^T = 0$. Other computations in algorithm 4.2, including vector additions and truncations, are applied to each low-rank matrix component of the iterates.

Algorithm 4.2: Low-rank preconditioned MINRES method

initialization: $V^{(0)} = 0$, $W^{(0)} = 0$, $W^{(1)} = 0$, $\gamma_0 = 0$. Choose $X^{(0)}$, compute $V^{(1)} = F - \mathscr{A}(X^{(0)})$. $P^{(1)} = \mathscr{M}^{-1}(V^{(1)})$, $\gamma_1 = \sqrt{\langle P^{(1)}, V^{(1)} \rangle}$. Set $\eta = \gamma_1$, $s_0 = s_1 = 0$, and $c_0 = c_1 = 1$.
for $j = 1, 2, \ldots$ **do**
  $P^{(j)} = P^{(j)} / \gamma_j$
  $\tilde{R}^{(j)} = \mathscr{A}(P^{(j)})$, $R^{(j)} = \mathcal{T}_{\text{rel}}(\tilde{R}^{(j)})$
  $\delta_j = \langle R^{(j)}, P^{(j)} \rangle$
  $\tilde{V}^{(j+1)} = R^{(j)} - (\delta_j / \gamma_j) V^{(j)} - (\gamma_j / \gamma_{j-1}) V^{(j-1)}$, $V^{(j+1)} = \mathcal{T}_{\text{rel}}(\tilde{V}^{(j+1)})$
  $P^{(j+1)} = \mathscr{M}^{-1}(V^{(j+1)})$
  $\gamma_{j+1} = \sqrt{\langle P^{(j+1)}, V^{(j+1)} \rangle}$
  $\alpha_0 = c_j \delta_j - c_{j-1} s_j \gamma_j$
  $\alpha_1 = \sqrt{\alpha_0^2 + \gamma_{j+1}^2}$
  $\alpha_2 = s_j \delta_j + c_{j-1} c_j \gamma_j$
  $\alpha_3 = s_{j-1} \gamma_j$
  $c_{j+1} = \alpha_0 / \alpha_1$, $s_{j+1} = \gamma_{j+1} / \alpha_1$
  $\tilde{W}^{(j+1)} = (P^{(j)} - \alpha_3 W^{(j-1)} - \alpha_2 W^{(j)}) / \alpha_1$, $W^{(j+1)} = \mathcal{T}_{\text{rel}}(\tilde{W}^{(j+1)})$
  $\tilde{X}^{(j)} = X^{(j-1)} + c_{j+1} \eta W^{(j+1)}$, $X^{(j)} = \mathcal{T}_{\text{rel}}(\tilde{X}^{(j)})$
  $\eta = -s_{j+1} \eta$
  Check convergence
end

4.5.2 Numerical experiments

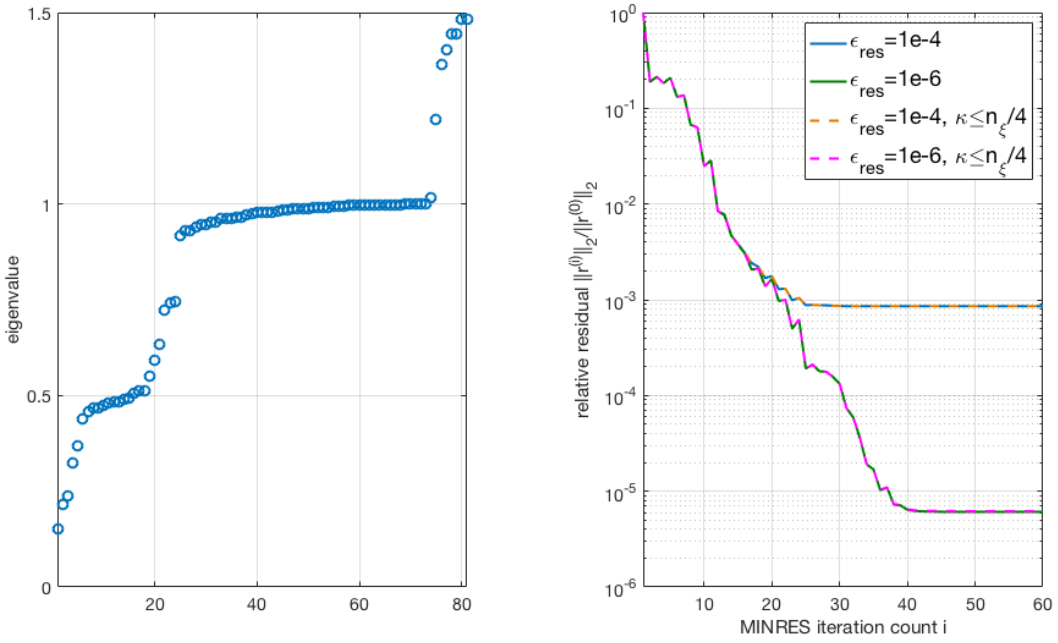
Consider a two-dimensional channel flow on domain $\mathcal{D} = [-1, 1]^2$ with uniform square meshes. Let $\partial\mathcal{D}_{\text{D}} = \{(x_1, x_2) \mid x_1 = -1, \text{ or } x_2 = 1, \text{ or } x_2 = -1\}$ and $\partial\mathcal{D}_{\text{N}} = \{(x_1, x_2) \mid x_1 = 1\}$. Define grid level n_c so that $2/h = 2^{n_c}$, where h is the mesh size. We use the Taylor–Hood method for finite element discretization with biquadratic basis functions $\{\vec{\phi}_i(x)\}$ for the velocity field and bilinear basis functions $\{\varphi_k(x)\}$ for the pressure field. For the velocity field the basis functions are in the

form $\left\{\begin{pmatrix}\phi_i(x)\\0\end{pmatrix}, \begin{pmatrix}0\\\phi_i(x)\end{pmatrix}\right\}$, where $\{\phi_i(x)\}$ are scalar-value biquadratic basis functions. The number of degrees of freedom in the spatial discretization is $n_x = n_u + n_p$ where $n_u = 2((2^{n_c+1}+1)^2 - n_{\partial\mathcal{D}_\mathrm{D}})$, $n_{\partial\mathcal{D}_\mathrm{D}}$ is the number of Dirichlet boundary nodes, and $n_p = (2^{n_c}+1)^2$. Assume the viscosity $a(x,\xi)$ has a KL expansion with the same specifications as in the diffusion problem. For the quadrature rule in section 4.3.2, we use a Smolyak sparse grid with Clenshaw–Curtis quadrature points and grid level 3.

We use stochastic inverse iteration algorithm to find the minimal eigenvalue of eq. (4.51). The eigenvalues of $BK_0^{-1}B^Tq = \lambda Mq$ are plotted in fig. 4.3a with $n_c = 3$. It shows that the minimal eigenvalue is isolated from the larger ones. For the inverse iteration, we take $\epsilon_\theta^{(i)}$ in eq. (4.30) as error indicator and use a stopping criterion $\epsilon_\theta^{(i)} \leq tol_{\mathrm{isi}} = 10^{-5}$. The error tolerance for the MINRES solver $tol_{\mathrm{minres}}^{(i)}$ is set as in eq. (4.44). Figure 4.3b shows the convergence of the low-rank MINRES method for different relative truncation tolerances ϵ_{rel}. It indicates the accuracy that MINRES can achieve is related to ϵ_{rel}. In the numerical experiments we use $\epsilon_{\mathrm{rel}}^{(i)} = 10^{-1} * tol_{\mathrm{minres}}^{(i)}$. In addition, we have observed that in many cases the truncations in Lines 4, 6, 14 of algorithm 4.2 produce relatively high ranks, which increases the computational cost. To handle this, we impose a bound on the ranks κ of the outputs of these truncation operators such that $\kappa \leq n_\xi/4$ (in general $n_x \geq n_\xi$). It is shown in Figure 4.3b that the convergence of low-rank MINRES is unaffected by this strategy.

Table 4.7 shows the ranks of the MINRES solutions $\mathbf{s}$ and $\hat{\mathbf{v}}^{(i)}$ in eq. (4.55) and numbers of MINRES steps it_{minres} required in each iteration. The solution matrices

S and $\hat{V}^{(i)}$ have sizes $n_u \times n_\xi$ and $n_p \times n_\xi$ (for $n_c = 4$ and $m = 11$, $n_u = 1984$, $n_p = 289$, $n_\xi = 364$), whereas their respective ranks are no larger than 31 and 50. In the Rayleigh quotient computation, the system eq. (4.56) is solved by a low-rank conjugate gradient method [56] with a relative residual smaller than 10^{-8}.



(a) (b)

Figure 4.3: (a) Eigenvalues of $BK_0^{-1}B^T q = \lambda M q$. $n_c = 3$. (b) Reduction of the relative residual for the low-rank MINRES method with various truncation criteria. Solid lines: relative tolerance ϵ_{rel}; dashed lines: relative tolerance ϵ_{rel} with rank $\kappa \leq n_\xi/4$. $n_c = 4$, $b = 4.0$, $m = 11$.

As in the diffusion problem, we show the accuracy of the low-rank stochastic Galerkin approach by comparing the results with the reference solutions from Monte Carlo simulations using `eigs`. Let $m = 11$, $p = 3$, $n_\xi = 364$. We use a sample size $n_r = 1000$. Table 4.8 shows the accuracy of the stochastic Galerkin solutions where ϵ_{λ^1} and ϵ_{u^1} are defined in eq. (4.45) (no Rayleigh–Ritz procedure is used here). In all cases, convergence of the inverse iteration takes 16–18 steps.

Table 4.7: Iterate ranks after the MINRES solve and numbers of MINRES steps required in the inverse iteration algorithm. $n_c = 4$, $b = 4.0$, $m = 11$.

(i)		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Rank	s	4	12	13	13	17	18	21	24	28	31	30	31	30	30	31	30
	$\hat{v}$	7	12	13	16	19	25	31	38	44	49	49	49	50	49	50	50
it_{minres}		22	35	35	37	37	39	39	41	42	43	43	43	43	43	43	43

Table 4.8: Relative difference between stochastic Galerkin solutions and Monte Carlo solutions. $b = 4.0$, $m = 11$, $n_\xi = 364$.

n_c	4	5	6
ϵ_{λ^1}	5.8903×10^{-9}	6.8722×10^{-9}	7.6883×10^{-9}
ϵ_{u^1}	4.4363×10^{-5}	5.1253×10^{-5}	5.3235×10^{-5}

As we did for the diffusion problem, we assess the the efficiency of the low-rank stochastic Galerkin method by comparison with the full-rank method and Monte Carlo simulation. For the latter, we use an LOBPCG solver preconditioned with the pressure mass matrix M, and the action of M^{-1} is again approximated by 10 steps of Chebyshev iteration. In this case, a stopping tolerance of 10^{-6} is used for LOBPCG to produce solutions with accuracy comparable to those obtained using the stochastic Galerkin approach. Table 4.9 shows the comparative costs of these methods when 1000 samples are used in a simulation. It is clear that the low-rank stochastic Galerkin method is more efficient than its full-rank counterpart, and the simulations using the surrogate solution obtained from the stochastic Galerkin approach are very cheap compared with Monte Carlo simulation. If we take the total cost of the stochastic Galerkin method to be the sum of t_{solve} and t_{sample},

then the comparison depends on the number of samples used, and in this measure, for 1000 samples it is cheaper to perform Monte Carlo simulation. This issue is explored in more detail in section 4.5.2, which interpolates the costs from timings using 1000, 5000 and 10000 samples and shows "crossover" sample sizes for which the stochastic Galerkin methods will be more efficient than Monte Carlo methods; these are approximately 2500 for the low-rank version and 4000 for the full-rank one.

Table 4.9: Time comparison (in seconds) between stochastic Galerkin method and Monte Carlo simulation for various n_c. $n_r = 1000$.

n_c		4	5	6
n_p		289	1089	4225
n_x		2273	9153	36737
low-rank SG	t_{solve}	269.84	1006.36	4382.16
	t_{sample}	0.11	0.13	0.25
full-rank SG	t_{solve}	324.74	1264.05	6272.26
	t_{sample}	0.11	0.22	0.97
MC		122.58	417.62	1594.47

(a) $b = 4.0$, $m = 11$, $n_\xi = 364$

n_c		4	5	6
n_p		289	1089	4225
n_x		2273	9153	36737
low-rank SG	t_{solve}	79.48	323.40	1557.33
	t_{sample}	0.06	0.07	0.09
full-rank SG	t_{solve}	132.63	538.44	2636.93
	t_{sample}	0.07	0.07	0.40
MC		128.16	411.34	1539.16

(b) $b = 5.0$, $m = 8$, $n_\xi = 165$

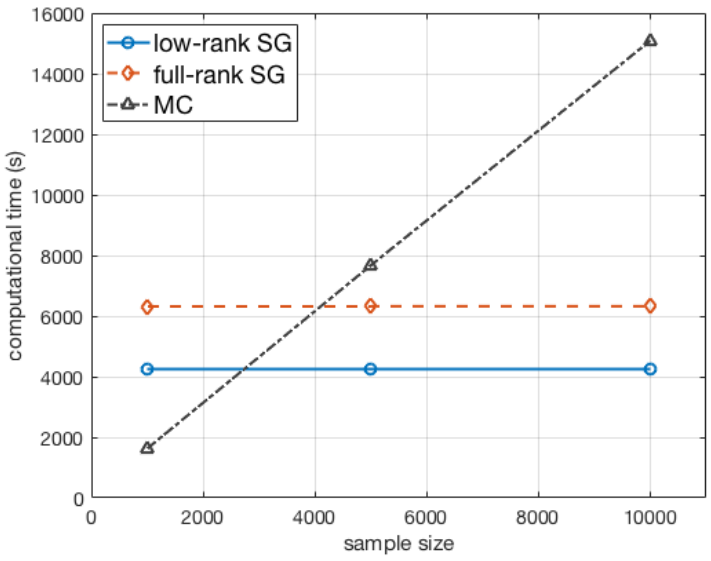


Figure 4.4: Computational time required by the low-rank stochastic Galerkin method, the full-rank stochastic Galerkin method, and the Monte Carlo method to generate large numbers of sample solutions. $nc = 6$, $b = 4.0$, $m = 11$, $n_r = 1000, 5000, 10000$.

4.6 Conclusions

We studied low-rank solution methods for the stochastic eigenvalue problems. The stochastic Galerkin approach was used to compute surrogate approximations to the minimal eigenvalues and corresponding eigenvectors, which are stochastic functions with gPC expansions. We introduced low-rank approximations to enhance efficiency of the stochastic inverse subspace iteration algorithm. Two detailed benchmark problems, the stochastic diffusion problem, and an operator associated with a discrete stochastic Stokes equation, were considered for illustrating the effectiveness of the proposed low-rank algorithm. It was confirmed in the numerical experiments that the low-rank solution method produces accurate results with much less computing time, making the stochastic Galerkin method more competitive com-

pared with the sample-based Monte Carlo approach.

Chapter 5: Low-rank solvers for the stochastic unsteady Navier–Stokes equations

5.1 Introduction

In this chapter, we develop some new computational methods for solving the stochastic unsteady Navier–Stokes equations, using stochastic Galerkin methods to address the stochastic nature of the problem and so-called all-at-once treatment of time integration.

For a time-dependent problem, the solutions at different time steps are usually computed in a sequential manner via time-stepping. For example, a fully-implicit scheme with adaptive time step sizes was studied in [24, 39, 49]. On the other hand, an all-at-once system can be formed by collecting the algebraic systems at all the discrete time steps into a single one, and the solutions are computed simultaneously. Such a formulation avoids the serial nature of time-stepping, and allows parallelization in the time direction for accelerating the solution procedure [31, 64, 66]. A drawback, however, is that for large-size problems, the all-at-once system may require excessive storage. In this chapter, we address this issue by using a low-rank tensor representation of data within the solution methods.

We develop a low-rank iterative algorithm for solving the unsteady Navier–Stokes equations with an uncertain viscosity. The equations are linearized with Picard's method. At each step of the nonlinear iteration, the stochastic Galerkin discretization gives rise to a large linear system, which is solved by a Krylov subspace method. Similar approaches have been used to study the steady-state problem [75, 87], where the authors also proposed effective preconditioners by taking advantage of the special structures of the linear systems. To reduce memory and computational costs, we compute low-rank approximations to the discrete solutions, which are represented as three-dimensional tensors in the all-at-once formulation. We refer to [38] for a review of low-rank tensor approximation techniques, and we will use the tensor train decomposition [70] in this work. The tensor train decomposition allows efficient basic operations on tensors. A truncation procedure is also available to compress low-rank tensors in the tensor train format to ones with smaller ranks.

Our goal is to use the low-rank tensors within Krylov subspace methods, in order to efficiently solve the large linear systems arising in each nonlinear step. The basic idea is to represent all the vector quantities that arise during the course of a Krylov subspace computation as low-rank tensors. With this strategy, much less memory is needed to store the data produced during the iteration. Moreover, the associated computations, such as matrix-vector products and vector additions, become much cheaper. The tensors are compressed in each iteration to maintain low ranks. This idea has been used for the conjugate gradient method and the GMRES method, with different low-rank tensor formats [2, 4, 18, 56]. In addition, the convergence of Krylov subspace methods can be greatly improved by an effective

preconditioner. In conjunction with the savings achieved through low-rank tensor computations, we will derive preconditioners for the stochastic all-at-once formulation based on some state-of-the-art techniques used for deterministic problems, and we will demonstrate their performances in numerical experiments. We also explore the idea of inexact Picard methods where the linear systems are solved inexactly at each Picard step to further save computational work, and we show that with this strategy very small numbers of iterations are needed for the Krylov subspace method.

We note that a different type of approach, the alternating iterative methods [19, 46, 81], including the density matrix renormalization group (DMRG) algorithm and its variants, can be used for solving linear systems in the tensor train format. In these methods, each component of the low-rank solution tensor is approached directly and optimized by projecting to a small local problem. This approach avoids the rank growth in intermediate iterates typically encountered in a low-rank Krylov subspace method. However, these methods are developed for solving symmetric positive definite systems and require nontrivial effort to be adapted for a nonsymmetric Navier–Stokes problem.

The rest of the chapter is organized as follows. In section 5.2 we give a formal presentation of the problem. Discretization techniques that result in an all-at-once linear system at each Picard step are discussed in section 5.3. In section 5.4 we introduce the low-rank tensor approximation and propose a low-rank Krylov subspace iterative solver for the all-at-once systems. The preconditioners are derived in section 5.5 and numerical results are given in section 5.6.

5.2 Problem setting

Consider the unsteady Navier–Stokes equations for incompressible flows on a space-time domain $\mathcal{D} \times (0, t_f]$,

$$\begin{aligned} \frac{\partial \vec{u}}{\partial t} - \nabla \cdot (\nu \nabla \vec{u}) + \vec{u} \cdot \nabla \vec{u} + \nabla p = \vec{0}, \\ \nabla \cdot \vec{u} = 0, \end{aligned} \tag{5.1}$$

where $\vec{u}$ and p stand for the velocity and pressure, respectively, ν is the viscosity, and $\mathcal{D}$ is a two-dimensional spatial domain with boundary $\partial\mathcal{D} = \partial\mathcal{D}_{\mathrm{D}} \cup \partial\mathcal{D}_{\mathrm{N}}$. The Dirichlet boundary $\partial\mathcal{D}_{\mathrm{D}}$ consists of an inflow boundary and fixed walls, and Neumann boundary conditions are set for the outflow,

$$\begin{aligned} \vec{u} = \vec{u}_{\mathrm{D}} \quad &\text{on } \partial\mathcal{D}_{\mathrm{D}}, \\ \nu \nabla \vec{u} \cdot \vec{n} - p\vec{n} = \vec{0} \quad &\text{on } \partial\mathcal{D}_{\mathrm{N}}. \end{aligned} \tag{5.2}$$

We assume the Neumann boundary $\partial\mathcal{D}_{\mathrm{N}}$ is not empty so that the pressure p is uniquely determined. The function $\vec{u}_{\mathrm{D}}(x, t)$ denotes a time-dependent inflow, typically growing from zero to a steady state, and it is set to zero at fixed walls. The initial conditions are zero everywhere for both $\vec{u}$ and p.

The uncertainty in the problem is introduced by a stochastic viscosity parameter ν, which is modeled as a random field depending on a finite collection of random variables $\{\xi_l\}_{l=1}^m$ (or written as a vector ξ). Specifically, we consider a representation as a truncated KL expansion,

$$\nu(x, \xi) = \nu_0(x) + \sum_{l=1}^{m} \nu_l(x)\xi_l, \tag{5.3}$$

where ν_0 is the mean viscosity, and $\{\nu_l\}_{l=1}^m$ are determined by the covariance function of ν. We assume the viscosity satisfies $\nu(x,\xi) \geq \nu_{\min} > 0$ almost surely for any $x \in \mathcal{D}$. We refer to [75, 87] for different forms of the stochastic viscosity. The solutions $\vec{u}$ and p in eq. (5.1) will also be random fields which depend on the space parameter x, time t, and the random variables ξ.

5.3 Discrete problem

In this section, we derive a fully discrete problem for the stochastic unsteady Navier–Stokes equations eq. (5.1). This involves a time discretization scheme and a stochastic Galerkin discretization for the physical and stochastic spaces at each time step. The discretizations give rise to a nonlinear algebraic system. Instead of solving such a system at each time step, we collect them to form an all-at-once system, where the discrete solutions at all the time steps are solved simultaneously. The discrete problem is then linearized with Picard's method, and a large linear system is solved at each step of the nonlinear iteration.

5.3.1 Time discretization

For simplicity we use the backward Euler method for time discretization, which is first-order accurate but unconditionally stable and dissipative. Divide the interval $(0, t_f]$ into n_t uniform steps $\{t_k\}_{k=1}^{n_t}$ with step size $\tau = t_f/n_t$ and initial time $t_0 = 0$. Given the solution at time t_{k-1}, we need to solve the following equations for $\vec{u}^k$ and

p^k:

$$\begin{aligned} \frac{\vec{u}^k - \vec{u}^{k-1}}{\tau} - \nabla \cdot (\nu \nabla \vec{u}^k) + \vec{u}^k \cdot \nabla \vec{u}^k + \nabla p^k &= \vec{0}, \\ \nabla \cdot \vec{u}^k &= 0. \end{aligned} \tag{5.4}$$

The implicit method requires solving an algebraic system at each time step. In the following we discuss how the system is assembled from the stochastic Galerkin discretization of eq. (5.4).

5.3.2 Stochastic Galerkin method

At time step k, the stochastic Galerkin method finds parametrized approximate velocity solutions $\vec{u}_h^k$ and pressure solutions p_h^k in finite-dimensional subspaces of $(H^1(\mathcal{D}))^2 \otimes L^2(\Gamma)$ and $L^2(\mathcal{D}) \otimes L^2(\Gamma)$, where Γ is the joint image of the random variables $\{\xi_l\}$. The functional spaces are defined as follows,

$$\begin{aligned} (H^1(\mathcal{D}))^2 \otimes L^2(\Gamma) &:= \Big\{ \vec{v} : \mathcal{D} \times \Gamma \to \mathbb{R} \mid \mathbb{E}\big[\|\vec{v}\|^2_{(H^1(\mathcal{D}))^2}\big] < \infty \Big\}, \\ L^2(\mathcal{D}) \otimes L^2(\Gamma) &:= \Big\{ q : \mathcal{D} \times \Gamma \to \mathbb{R} \mid \mathbb{E}\big[\|q\|^2_{L^2(\mathcal{D})}\big] < \infty \Big\}. \end{aligned} \tag{5.5}$$

The expectations are taken with respect to the distribution of random variables $\{\xi_l\}$. In the following we use $\langle \cdot \rangle$ to denote the expected value. Let the finite-dimensional subspaces be $\mathcal{X} = \text{span}\{\vec{\phi}_i(x)\} \subset (H^1(\mathcal{D}))^2$, $\mathcal{Y} = \text{span}\{\varphi_i(x)\} \subset L^2(\mathcal{D})$, and $\mathcal{Z} = \text{span}\{\psi_r(\xi)\} \subset L^2(\Gamma)$. Let $\mathcal{X}_{\text{D}}^k$ and $\mathcal{X}_0$ be the spaces of functions in $\mathcal{X}$ with Dirichlet boundary conditions $\vec{u}_{\text{D}}(x, t_k)$ and $\vec{0}$ imposed for the velocity field, respectively. Then for eq. (5.4) the stochastic Galerkin formulation entails the computation of

$\vec{u}_h^k \in \mathcal{X}_{\mathrm{D}}^k \otimes \mathcal{Z}$ and $p_h^k \in \mathcal{Y} \otimes \mathcal{Z}$, satisfying the weak form

$$\begin{aligned} \tau^{-1}\langle(\vec{u}_h^k, \vec{v}_h)\rangle - \tau^{-1}\langle(\vec{u}_h^{k-1}, \vec{v}_h)\rangle + \langle(\nu\nabla\vec{u}_h^k, \nabla\vec{v}_h)\rangle \\ + \langle(\vec{u}_h^k \cdot \nabla\vec{u}_h^k, \vec{v}_h)\rangle - \langle(p_h^k, \nabla\cdot\vec{v}_h)\rangle = 0, \\ \langle(\nabla\cdot\vec{u}_h^k, q_h)\rangle = 0, \end{aligned} \tag{5.6}$$

for any $\vec{v}_h \in \mathcal{X}_0 \otimes \mathcal{Z}$ and $q_h \in \mathcal{Y} \otimes \mathcal{Z}$. Here, $(\cdot,\cdot)$ means the inner product in $L^2(\mathcal{D})$. For the physical spaces, we use a stable Taylor–Hood discretization [26] on quadrilateral elements, with biquadratic basis functions $\{\vec{\phi}_i\}_{i=1}^{n_u} = \left\{\begin{pmatrix}\phi_i\\0\end{pmatrix}, \begin{pmatrix}0\\\phi_i\end{pmatrix}\right\}_{i=1}^{n_u/2}$ for velocity, and bilinear basis functions $\{\varphi_i\}_{i=1}^{n_p}$ for pressure. The stochastic basis functions $\{\psi_r\}_{r=1}^{n_\xi}$ are m-dimensional orthonormal polynomials constructed from generalized polynomial chaos (gPC, [96]) satisfying $\langle\psi_r\psi_s\rangle = \delta_{rs}$. The stochastic Galerkin solutions are expressed as linear combinations of the basis functions,

$$\begin{aligned} \vec{u}_h^k(x,\xi) &= \sum_{s=1}^{n_\xi}\sum_{j=1}^{n_u} u_{js}^k \vec{\phi}_j(x)\psi_s(\xi), \\ p_h^k(x,\xi) &= \sum_{s=1}^{n_\xi}\sum_{j=1}^{n_p} p_{js}^k \varphi_j(x)\psi_s(\xi). \end{aligned} \tag{5.7}$$

The coefficient vectors $\boldsymbol{u}^k = [u_{11}^k, u_{21}^k, \ldots, u_{n_u 1}^k, \ldots, u_{1n_\xi}^k, u_{2n_\xi}^k, \ldots, u_{n_u n_\xi}^k]$ and similarly defined $\boldsymbol{p}^k$ are computed from the nonlinear algebraic system

$$\begin{pmatrix} \mathbb{F}^k(\boldsymbol{u}) & I_{n_\xi}\otimes B^T \\ I_{n_\xi}\otimes B & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{u}^k \\ \boldsymbol{p}^k \end{pmatrix} + \begin{pmatrix} -\tau^{-1}(I_{n_\xi}\otimes \boldsymbol{M}) & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{u}^{k-1} \\ \boldsymbol{p}^{k-1} \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}^{u,k} \\ \boldsymbol{f}^{p,k} \end{pmatrix} \tag{5.8}$$

where

$$\mathbb{F}^k(\boldsymbol{u}) = \tau^{-1}(I_{n_\xi}\otimes\boldsymbol{M}) + \sum_{l=0}^{m}(G_l\otimes\boldsymbol{A}_l) + \sum_{l=1}^{n_\xi}(H_l\otimes\boldsymbol{N}(\vec{u}_{h,l}^k)). \tag{5.9}$$

Here I_{n_ξ} is the $n_\xi \times n_\xi$ identity matrix, and $\otimes$ means the Kronecker product of two matrices. The boldface matrices $\boldsymbol{M}$, $\boldsymbol{A}_l$, and $\boldsymbol{N}(\vec{u}_{h,l}^k)$ are 2×2 block-diagonal, with

the scalar mass matrix M, weighted stiffness matrix A_l, and discrete convection operator $N(\vec{u}_{h,l}^k)$ as diagonal components, where

$$[M]_{ij} = (\phi_j, \phi_i),\ [A_l]_{ij} = (\nu_l \nabla\phi_j, \nabla\phi_i),\ [N(\vec{u}_{h,l}^k)]_{ij} = (\vec{u}_{h,l}^k \cdot \nabla\phi_j, \phi_i), \tag{5.10}$$

for $i, j = 1, \ldots, n_u/2$. Note the dependency on $\boldsymbol{u}^k$ comes from the nonlinear convection term $\boldsymbol{N}$, with convection velocity $\vec{u}_{h,l}^k = \sum_j u_{jl}^k \vec{\phi}_j(x)$. Let $x = (x_1, x_2)$. The discrete divergence operator $B = [B_{x_1}, B_{x_2}]$, with

$$[B_{x_1}]_{ij} = -(\varphi_i, \frac{\partial\phi_j}{\partial x_1}),\ [B_{x_2}]_{ij} = -(\varphi_i, \frac{\partial\phi_j}{\partial x_2}), \tag{5.11}$$

for $i = 1, \ldots, n_p$ and $j = 1, \ldots, n_u/2$. The matrices $\{G_l\}_{l=0}^m$ and $\{H_l\}_{l=1}^{n_\xi}$ of eq. (5.9) come from the stochastic basis functions and have entries

$$[G_l]_{rs} = \langle \xi_l \psi_r \psi_s \rangle,\ [H_l]_{rs} = \langle \psi_l \psi_r \psi_s \rangle, \tag{5.12}$$

for $r, s = 1, \ldots, n_\xi$, where $\xi_0 \equiv 1$. These matrices are also sparse due to orthogonality of the basis functions [29]. The Dirichlet boundary conditions are incorporated in the right-hand side of eq. (5.8).

5.3.3 All-at-once system

As discussed in the beginning of the section, we consider an all-at-once system where the discrete solutions at all the time steps are computed together. Let

$$\boldsymbol{u} = \begin{pmatrix} \boldsymbol{u}^1 \\ \boldsymbol{u}^2 \\ \vdots \\ \boldsymbol{u}^{n_t} \end{pmatrix} \in \mathbb{R}^{n_t n_\xi n_u} \tag{5.13}$$

and let $\boldsymbol{p}$, $\boldsymbol{f}^u$, and $\boldsymbol{f}^p$ be similarly defined. By collecting the algebraic systems eq. (5.8) corresponding to all the time steps $\{t_k\}_{k=1}^{n_t}$, we get the single system

$$\begin{pmatrix} \mathbb{F}(\boldsymbol{u}) + \mathbb{C} & \mathbb{B}^T \\ \mathbb{B} & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{u} \\ \boldsymbol{p} \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}^u \\ \boldsymbol{f}^p \end{pmatrix}, \tag{5.14}$$

where $\mathbb{F}(\boldsymbol{u})$ is block diagonal with $\mathbb{F}^k(\boldsymbol{u})$ as the kth diagonal block, $\mathbb{B} = I_{n_t} \otimes I_{n_\xi} \otimes B$, and $\mathbb{C} = -\tau^{-1} C_{n_t} \otimes I_{n_\xi} \otimes \boldsymbol{M}$ with $C_{n_t} = \begin{pmatrix} 0 & & & \\ 1 & 0 & & \\ & \ddots & \ddots & \\ & & 1 & 0 \end{pmatrix} \in \mathbb{R}^{n_t \times n_t}$. Note that the zero initial conditions are incorporated in eq. (5.8) for $k = 1$. The all-at-once system eq. (5.14) is nonsymmetric and blockwise sparse. Each part of the system contains sums of Kronecker products of three matrices, i.e., in the form $\sum_l X_l^{(1)} \otimes X_l^{(2)} \otimes X_l^{(3)}$. In fact, from eq. (5.9),

$$\mathbb{F}(\boldsymbol{u}) = \tau^{-1} I_{n_t} \otimes I_{n_\xi} \otimes \boldsymbol{M} + \sum_{l=0}^{m} (I_{n_t} \otimes G_l \otimes \boldsymbol{A}_l) + \mathbb{N}(\boldsymbol{u}). \tag{5.15}$$

We discuss later (see section 5.4.3) how the convection matrix $\mathbb{N}$ can also be put in the Kronecker product form. It will be seen that this structure is useful for efficient matrix-vector product computations.

5.3.4 Picard's method

We use Picard's method to solve the nonlinear equation eq. (5.14). Picard's method is a fixed-point iteration. Let $\boldsymbol{u}^{(i)}$, $\boldsymbol{p}^{(i)}$ be the approximate solutions at the ith step. Each Picard step entails solving a large linear system

$$\begin{pmatrix} \mathbb{F}(\boldsymbol{u}^{(i-1)}) + \mathbb{C} & \mathbb{B}^T \\ \mathbb{B} & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{u}^{(i)} \\ \boldsymbol{p}^{(i)} \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}^u \\ \boldsymbol{f}^p \end{pmatrix}. \tag{5.16}$$

Instead of eq. (5.16), one can equivalently solve the corresponding residual equation for a correction of the solution. Let $\boldsymbol{u}^{(i)} = \boldsymbol{u}^{(i-1)} + \delta\boldsymbol{u}^{(i)}$, $\boldsymbol{p}^{(i)} = \boldsymbol{p}^{(i-1)} + \delta\boldsymbol{p}^{(i)}$. Then $\delta\boldsymbol{u}^{(i)}$ and $\delta\boldsymbol{p}^{(i)}$ satisfy

$$\begin{pmatrix} \mathbb{F}(\boldsymbol{u}^{(i-1)}) + \mathbb{C} & \mathbb{B}^T \\ \mathbb{B} & 0 \end{pmatrix} \begin{pmatrix} \delta\boldsymbol{u}^{(i)} \\ \delta\boldsymbol{p}^{(i)} \end{pmatrix} = \begin{pmatrix} \boldsymbol{r}^{u,(i-1)} \\ \boldsymbol{r}^{p,(i-1)} \end{pmatrix}, \tag{5.17}$$

where the nonlinear residual

$$\boldsymbol{r}^{(i)} = \begin{pmatrix} \boldsymbol{r}^{u,(i)} \\ \boldsymbol{r}^{p,(i)} \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}^u \\ \boldsymbol{f}^p \end{pmatrix} - \begin{pmatrix} \mathbb{F}(\boldsymbol{u}^{(i)}) + \mathbb{C} & \mathbb{B}^T \\ \mathbb{B} & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{u}^{(i)} \\ \boldsymbol{p}^{(i)} \end{pmatrix}. \tag{5.18}$$

The complete algorithm is summarized in algorithm 5.1. The initial iterates $\boldsymbol{u}^{(0)}$, $\boldsymbol{p}^{(0)}$ are given by solving a Stokes problem, for which in eq. (5.16) the convection matrix $\mathbb{N}$ is set to zero.

Algorithm 5.1 Picard's method

1: Solve Stokes problem for initial $\boldsymbol{u}^{(0)}$, $\boldsymbol{p}^{(0)}$, update convection matrix $\mathbb{N}(\boldsymbol{u}^{(0)})$, and compute nonlinear residual $\boldsymbol{r}^{(0)}$. $i = 0$.
2: **while** $\|\boldsymbol{r}^{(i)}\|_2 > tol * \|\boldsymbol{r}^{(0)}\|_2$ and $i < maxit$ **do**
3: $\quad i = i + 1$
4: $\quad$ Solve linear system eq. (5.17) for $\delta\boldsymbol{u}^{(i)}$, $\delta\boldsymbol{p}^{(i)}$
5: $\quad$ Update solution $\boldsymbol{u}^{(i)}$, $\boldsymbol{p}^{(i)}$
6: $\quad$ Update convection matrix $\mathbb{N}(\boldsymbol{u}^{(i)})$
7: $\quad$ Compute nonlinear residual $\boldsymbol{r}^{(i)}$
8: **end while**
9: **return** $\boldsymbol{u}^{(i)}$, $\boldsymbol{p}^{(i)}$

5.4 Low-rank approximation

In this section we discuss low-rank approximation techniques and how they can be used with iterative solvers. The computational cost of solving eq. (5.17) at each Picard step is high due to the large problem size $n_t n_\xi (n_u + n_p)$, especially when

large numbers of spatial grid points or time steps are used to achieve high-resolution solutions. We will address this using low-rank tensor approximations to the solution vectors $\boldsymbol{u}$ and $\boldsymbol{p}$. We will develop efficient iterative solvers and preconditioners where the solution is approximated using a compressed data representation in order to greatly reduce the memory requirement and computational effort. The idea is to represent the iterates in a Krylov subspace method in a low-rank tensor format. The basic operations associated with the low-rank format are much cheaper, and as the Krylov subspace method converges it constructs a sequence of low-rank approximations to the solution of the system.

5.4.1 Tensor train decomposition

A tensor $\underline{z} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ is a multidimensional array with entries $\underline{z}(i_1, \ldots, i_d)$, where $i_l = 1, \ldots, n_l$, $l = 1, \ldots, d$. The solution coefficients in eq. (5.7) can be represented in the form of three-dimensional $n_t \times n_\xi \times n_x$ tensors $\underline{u}$ (where $n_x = n_u$) and $\underline{p}$ ($n_x = n_p$), such that $\underline{u}(k, s, j) = u_{js}^k$ and $\underline{p}(k, s, j) = p_{js}^k$. Equivalently, such tensors can be represented in vector format, where the vector version $\boldsymbol{u}$ and $\boldsymbol{p}$ are specified using the vectorization operation

$$\boldsymbol{u} = \mathrm{vec}(\underline{u}) \;\Leftrightarrow\; \boldsymbol{u}(\overline{i_1 i_2 i_3}) = \underline{u}(i_1, i_2, i_3) \tag{5.19}$$

where $\overline{i_1 i_2 i_3} = i_3 + (i_2 - 1)n_x + (i_1 - 1)n_\xi n_x$, and $\boldsymbol{p} = \mathrm{vec}(\underline{p})$ in a similar manner. In an iterative solver for the system eq. (5.17), any iterate $\boldsymbol{z}$ can be equivalently represented as a three-dimensional tensor $\underline{z} \in \mathbb{R}^{n_t \times n_\xi \times n_x}$. In the sequel we use vector $\boldsymbol{z}$ and tensor $\underline{z}$ interchangebly. The tensor train decomposition [70] is a compressed

low-rank representation to approximate a given tensor and efficiently perform tensor operations. Specifically, the tensor train format of $\underline{z}$ is defined as

$$\underline{z}(i_1, i_2, i_3) \approx \sum_{\alpha_1, \alpha_2} \underline{z}^{(1)}(i_1, \alpha_1)\underline{z}^{(2)}(\alpha_1, i_2, \alpha_2)\underline{z}^{(3)}(\alpha_2, i_3), \tag{5.20}$$

where $\underline{z}^{(1)} \in \mathbb{R}^{n_t \times \kappa_1}$, $\underline{z}^{(2)} \in \mathbb{R}^{\kappa_1 \times n_\xi \times \kappa_2}$, $\underline{z}^{(3)} \in \mathbb{R}^{\kappa_2 \times n_x}$ are the tensor train cores, and κ_1 and κ_2 are called the tensor train ranks. It is easy to see that if $\kappa_1, \kappa_2 \approx \kappa$, the memory cost to store $\underline{z}$ is reduced from $O(n_t n_\xi n_x)$ to $O((n_t + n_\xi \kappa + n_x)\kappa)$.

The tensor train decomposition allows efficient basic operations on tensors. Most importantly, matrix-vector products can be computed much less expensively if the vector $\boldsymbol{z}$ is in the tensor train format. For $\underline{z}$ as in eq. (5.20), the vector $\boldsymbol{z}$ has an equivalent Kronecker product form [19]

$$\boldsymbol{z} = \mathrm{vec}(\underline{z}) = \sum_{\alpha_1, \alpha_2} z^{(1)}_{\alpha_1} \otimes z^{(2)}_{\alpha_1, \alpha_2} \otimes z^{(3)}_{\alpha_2}, \tag{5.21}$$

where in the right-hand side $z^{(1)}_{\alpha_1}$, $z^{(2)}_{\alpha_1,\alpha_2}$, and $z^{(3)}_{\alpha_2}$ are vectors of length n_t, n_ξ, and n_x, respectively, obtained by fixing the indices α_1 and α_2 in $\underline{z}^{(1)}$, $\underline{z}^{(2)}$, and $\underline{z}^{(3)}$. Then for any matrix $\mathbb{X} = X^{(1)} \otimes X^{(2)} \otimes X^{(3)}$, such as the blocks in eq. (5.17),

$$\mathbb{X}\boldsymbol{z} = \sum_{\alpha_1, \alpha_2} (X^{(1)} z^{(1)}_{\alpha_1}) \otimes (X^{(2)} z^{(2)}_{\alpha_1,\alpha_2}) \otimes (X^{(3)} z^{(3)}_{\alpha_2}). \tag{5.22}$$

The product is also in tensor train format with the same ranks as in $\boldsymbol{z}$ (of the right-hand side of eq. (5.20)), and it only requires matrix-vector products for each component of $\mathbb{X}$. The component matrices from eq. (5.15) are sparse with numbers of nonzeros proportional to n_t, n_ξ, and n_x, respectively, and the computational cost is thus reduced from $O(n_t n_\xi n_x)$ to $O((n_t + n_\xi \kappa + n_x)\kappa)$.

Other vector computations, including additions and inner products, are also inexpensive with the tensor train format. One thing to note is that the additions of two vectors in tensor train format will tend to increase the ranks. This can be easily seen from eq. (5.20), since the addition of two low-rank tensors end up with more terms for the summation on the right-hand side. An important operation for the tensor train format is the truncation (or rounding) operation, used to reduce the ranks for tensors that are already in the tensor train format but have suboptimal high ranks. For a given tensor $\underline{z}$ as in eq. (5.20), the truncation computes

$$\tilde{\underline{z}} = \mathcal{T}_\epsilon(\underline{z}), \tag{5.23}$$

such that $\tilde{\underline{z}}$ has smaller ranks than $\underline{z}$ and satisfies the relative error $\|\tilde{\underline{z}} - \underline{z}\|_F / \|\underline{z}\|_F \leq \epsilon$. (Note that $\|\underline{z}\|_F = \|\boldsymbol{z}\|_2$.) The truncation operator is based on the TT-SVD algorithm [70], given in algorithm 5.2, which is used to compute a low-rank tensor train approximation for a full tensor $\underline{z} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$. In the algorithm, a sequence of singular value decompositions (SVDs) are computed for the so-called unfolding matrix Z, obtained by reshaping the entries of a tensor into a two-dimensional array. Terms corresponding to small singular values are dropped such that the error $\|E\|_F \leq \delta_j$, $j = 1, \ldots, d-1$ (see line 4 of algorithm 5.2). It was shown in [70] that the algorithm produces a tensor train $\tilde{\underline{z}}$ which satisfies

$$\|\underline{z} - \tilde{\underline{z}}\|_F \leq \Big(\sum_{k=1}^{d-1} \delta_k^2 \Big)^{1/2}. \tag{5.24}$$

Thus, one can choose $\delta_1 = \cdots = \delta_{d-1} = \epsilon \|\underline{z}\|_F / \sqrt{d-1}$ to achieve the relative error $\|\tilde{\underline{z}} - \underline{z}\|_F / \|\underline{z}\|_F \leq \epsilon$. Note the algorithm is costly since it requires SVDs on matrices $Z \in \mathbb{R}^{\kappa_{j-1} n_j \times n_{j+1} \cdots n_d}$. However, when the tensor $\underline{z}$ is already in the tensor

train format, the computation can be greatly simplified, and only SVDs on the much smaller tensor train cores are needed. The cost of the truncation operation is $O(dn\kappa^3)$ if $n_1, \ldots, n_d \approx n$ and $\kappa_1, \ldots, \kappa_{d-1} \approx \kappa$. We refer to [70] for more details. In the numerical experiments, we use TT-Toolbox [71] for tensor train computations.

Algorithm 5.2 TT-SVD

Let $Z = \underline{z}$. Set truncation parameters $\{\delta_j\}$. $\kappa_0 = 1$.
for $j = 1, \ldots, d-1$ **do**
  $Z \leftarrow \text{reshape}(Z, [\kappa_{j-1} n_j, n_{j+1} \cdots n_d])$
  Compute truncated SVD $Z = U\Sigma V^T + E$, $\|E\|_F \leq \delta_j$, $\kappa_j = \text{rank}(\Sigma)$
  New core $\tilde{\underline{z}}^{(j)} \leftarrow \text{reshape}(U, [\kappa_{j-1}, n_j, \kappa_j])$
  Update $Z \leftarrow \Sigma V^T$
end for
New core $\tilde{\underline{z}}^{(d)} \leftarrow Z$
return $\tilde{\underline{z}}$ in tensor train format with cores $\{\tilde{\underline{z}}^{(j)}\}$

5.4.2 Low-rank GMRES

The tensor train decomposition offers efficient tensor operations and we use it in iterative solvers to reduce the cost of the computations. The all-at-once system eq. (5.17) to be solved at each step of Picard's method is nonsymmetric. We use a right-preconditioned GMRES method to solve the system. The complete algorithm for solving $\mathscr{L}\boldsymbol{z} = \boldsymbol{f}$ is summarized in algorithm 5.3. The preconditioner $\mathscr{P}^{-1}$ entails an inner iterative process and is not exactly the same for each GMRES iteration, and therefore a variant of the flexible GMRES method (see, e.g., [**?**]) is used. As discussed above, all the iterates in the algorithm are represented in the tensor train format for efficient computations, and the truncation operation is used to compress the tensor train ranks so that they stay small relative to the problem size. It should

be noted that since the quantities are truncated, the Arnoldi vectors $\{\boldsymbol{v}_i\}$ do not form orthogonal basis for the Krylov subspace, and thus this is not a true GMRES computation. In section 5.5, we construct effective preconditioners for the system eq. (5.17).

Algorithm 5.3 Low-rank GMRES method

Choose initial $\boldsymbol{z}_0$, compute $\boldsymbol{r}_0 = \mathcal{T}_\epsilon(\boldsymbol{f} - \mathscr{L}\boldsymbol{z}_0)$, $\beta = \|\boldsymbol{r}_0\|_2$, and $\boldsymbol{v}_1 = \boldsymbol{r}_0/\beta$. Let $k = 0$.
while $\|\boldsymbol{r}_k\|_2 > tol * \|\boldsymbol{f}\|_2$ and $k < maxit$ **do**
  $k = k + 1$
  Compute $\hat{\boldsymbol{v}}_k = \mathscr{P}^{-1}\boldsymbol{v}_k$
  Compute $\boldsymbol{s} = \mathcal{T}_\epsilon(\mathscr{L}\hat{\boldsymbol{v}}_k)$
  for $i = 1, \ldots, k$ **do**
    $h_{ik} = \boldsymbol{s}^T\boldsymbol{v}_i$
    $\boldsymbol{s} = \boldsymbol{s} - h_{ik}\boldsymbol{v}_i$
  end for
  $h_{k+1,k} = \|\boldsymbol{s}\|_2$, $\boldsymbol{v}_{k+1} = \mathcal{T}_\epsilon(\boldsymbol{s}/h_{k+1,k})$
  Define $\hat{V}_k = [\hat{\boldsymbol{v}}_1, \ldots, \hat{\boldsymbol{v}}_k]$ and $\bar{H} \in \mathbb{R}^{(k+1)\times k}$ with $\bar{H}_{ij} = h_{ij}$
  Compute $y_k = \text{argmin}_y \|\beta e_1 - \bar{H}y\|_2$, where $e_1 = [1, 0, \ldots, 0]^T$
  Compute $\boldsymbol{z}_k = \mathcal{T}_\epsilon(\boldsymbol{z}_0 + \hat{V}_k y_k)$
  Compute $\boldsymbol{r}_k = \mathcal{T}_\epsilon(\boldsymbol{f} - \mathscr{L}\boldsymbol{z}_k)$
end while
return $\boldsymbol{z}_k$

5.4.3 Convection matrix

We now show that in eq. (5.15) if the velocity $\boldsymbol{u}$ is in the tensor train format, the convection matrix $\mathbb{N}(\boldsymbol{u})$ can be represented as a sum of Kronecker products of matrices [6], which allows efficient matrix-vector product computations as in eq. (5.22). Assume the coefficient tensor in eq. (5.7) is approximated by a tensor train decomposition,

$$u_{jl}^k = \underline{u}(k, l, j) = \sum_{\alpha_1, \alpha_2} \underline{u}^{(1)}(k, \alpha_1)\underline{u}^{(2)}(\alpha_1, l, \alpha_2)\underline{u}^{(3)}(\alpha_2, j). \tag{5.25}$$

Note the entries of $\boldsymbol{N}(\vec{u}^k_{h,l})$ are linear in $\vec{u}^k_{h,l}$ and

$$\vec{u}^k_{h,l} = \sum_j u^k_{jl}\vec{\phi}_j(x) = \sum_{\alpha_1,\alpha_2} \underline{u}^{(1)}(k,\alpha_1)\underline{u}^{(2)}(\alpha_1,l,\alpha_2)(\sum_j \underline{u}^{(3)}(\alpha_2,j)\vec{\phi}_j(x)). \tag{5.26}$$

Let $\vec{u}^{(3)}_{\alpha_2} = \sum_j \underline{u}^{(3)}(\alpha_2,j)\vec{\phi}_j(x)$. Then the kth diagonal block of $\mathbb{N}(\boldsymbol{u})$ is

$$\sum_{l=1}^{n_\xi}(H_l \otimes \boldsymbol{N}(\vec{u}^k_{h,l})) = \sum_{\alpha_1,\alpha_2} \underline{u}^{(1)}(k,\alpha_1)\sum_{l=1}^{n_\xi}(\underline{u}^{(2)}(\alpha_1,l,\alpha_2)H_l) \otimes \boldsymbol{N}(\vec{u}^{(3)}_{\alpha_2}). \tag{5.27}$$

The convection matrix $\mathbb{N}(\boldsymbol{u})$ can be expressed as

$$\mathbb{N}(\boldsymbol{u}) = \sum_{\alpha_1,\alpha_2} \mathrm{diag}(u^{(1)}_{\alpha_1}) \otimes \sum_{l=1}^{n_\xi}(\underline{u}^{(2)}(\alpha_1,l,\alpha_2)H_l) \otimes \boldsymbol{N}(\vec{u}^{(3)}_{\alpha_2}). \tag{5.28}$$

Here $u^{(1)}_{\alpha_1}$ is a vector obtained by fixing the index α_1 in $\underline{u}^{(1)}$, and $\mathrm{diag}(u^{(1)}_{\alpha_1})$ is a diagonal matrix with $u^{(1)}_{\alpha_1}$ on the diagonal. The result is a sum of Kronecker products of three smaller matrices. Such a representation can be constructed for any iterate $\boldsymbol{u}^{(i)}$ in the tensor train format. However, given the number of terms in the summation in the right-hand side of eq. (5.28), the matrix-vector product with $\mathbb{N}$ will result in a significant tensor train rank increase from κ to κ^2. In section 5.6.2 we discuss how an approximation to $\mathbb{N}$ can be constructed in Picard's method to alleviate the rank increase. Also, in the iterative algorithm a truncation operation is applied after each matrix-vector product $\mathscr{L}\boldsymbol{z}$ to compress the tensor ranks.

5.5 Preconditioning

In this section we discuss preconditioning techniques for the all-at-once system eq. (5.17) so that the Krylov subspace methods converge in a small number of iterations. To simplify the notation, we use $\boldsymbol{w}$ instead of $\boldsymbol{u}^{(i-1)}$, and the associated

approximate solution at the kth time step is

$$\vec{w}_h^k(x,\xi) = \sum_{l=1}^{n_\xi} \sum_{j=1}^{n_u} w_{jl}^k \vec{\phi}_j(x)\psi_l(\xi) \tag{5.29}$$

with $\vec{w}_{h,l}^k = \sum_j w_{jl}^k \vec{\phi}_j(x)$. In the following the dependence on $\boldsymbol{w}$ in $\mathbb{F}(\boldsymbol{w})$ is omitted in most cases. We derive a preconditioner by extending ideas for more standard problems [26], starting with an "idealized" block triangular preconditioner

$$\mathscr{P} = \begin{pmatrix} \mathbb{F}+\mathbb{C} & \mathbb{B}^T \\ 0 & -\mathbb{S} \end{pmatrix}. \tag{5.30}$$

With this choice of preconditioner, the Schur complement is $\mathbb{S} = \mathbb{B}(\mathbb{F}+\mathbb{C})^{-1}\mathbb{B}^T$, and the idealized preconditioned system derived from a block factorization

$$\begin{pmatrix} \mathbb{F}+\mathbb{C} & \mathbb{B}^T \\ \mathbb{B} & 0 \end{pmatrix} \mathscr{P}^{-1} = \begin{pmatrix} \mathbb{I} & 0 \\ \mathbb{B}\mathbb{F}^{-1} & \mathbb{I} \end{pmatrix} \tag{5.31}$$

has eigenvalues equal to 1, and Jordan blocks of order 2. Thus the right-preconditioned GMRES method will converge in two iterations. However, the application of $\mathscr{P}^{-1}$ involves solving linear systems associated with $\mathbb{S}$ and $\mathbb{F}+\mathbb{C}$. These are too expensive for practical computation and to develop preconditioners we will construct inexpensive approximations to the linear solves. Specifically, we derive mean-based preconditioners that use results from the mean deterministic problem. Such preconditioners for the stochastic steady-state Navier–Stokes equations have been studied in [75]. We generalize the techniques for the all-at-once formulation of the unsteady equations.

5.5.1 Deterministic operator

We review the techniques used for approximating the Schur complement in the deterministic case [26]. The approximations are based on the fact that a commutator of the convection-diffusion operator with the divergence operator

$$\mathcal{E} = \nabla \cdot (-\nu\nabla^2 + \vec{w}_{h,1}^k \cdot \nabla) - (-\nu\nabla^2 + \vec{w}_{h,1}^k \cdot \nabla)_p \nabla \cdot \tag{5.32}$$

is small under certain assumptions about smoothness and boundary conditions. The subscript p means the operators are defined on the pressure space. For a discrete convection-diffusion operator $\boldsymbol{F} = \boldsymbol{A}_0 + \boldsymbol{N}(\vec{w}_{h,1}^k)$ (which is part of the mean problem we discuss later), as defined in eq. (5.10), an approximation to the Schur complement $S = B\boldsymbol{F}^{-1}B^T$ is identified from a discrete analogue of eq. (5.32),

$$E = (M_p^{-1}B)(\boldsymbol{M}^{-1}\boldsymbol{F}) - (M_p^{-1}F_p)(M_p^{-1}B) \approx 0, \tag{5.33}$$

where the subscript p means the corresponding matrices constructed on the pressure space. Equation (5.33) leads to an approximation to the Schur complement matrix,

$$S = B\boldsymbol{F}^{-1}B^T \approx M_p F_p^{-1} B \boldsymbol{M}^{-1} B^T. \tag{5.34}$$

The pressure convection-diffusion (PCD) preconditioner is constructed by replacing the mass matrices with approximations containing only their diagonal entries (denoted by a subscript $*$) in eq. (5.34),

$$S_{\text{PCD}}^{-1} = (B\boldsymbol{M}_*^{-1}B^T)^{-1} F_p M_{p*}^{-1}. \tag{5.35}$$

The least-squares commutator (LSC) preconditioner avoids the construction of matrices on the pressure space, with the approximation to F_p,

$$F_p \approx (B\boldsymbol{M}^{-1}\boldsymbol{F}\boldsymbol{M}^{-1}B^T)(B\boldsymbol{M}^{-1}B^T)^{-1}M_p \tag{5.36}$$

(see [26, section 9.2] for a derivation). The LSC preconditioner is obtained by substituting F_p in eq. (5.34) and replacing the mass matrices with their diagonals,

$$S_{\text{LSC}}^{-1} = (B\boldsymbol{M}_*^{-1}B^T)^{-1}(B\boldsymbol{M}_*^{-1}\boldsymbol{F}\boldsymbol{M}_*^{-1}B^T)(B\boldsymbol{M}_*^{-1}B^T)^{-1}. \tag{5.37}$$

For both preconditioners, the only use of the matrices $\boldsymbol{F}$ and F_p is through matrix-vector products with them.

5.5.2 Approximations to $\mathbb{S}^{-1}$

The Schur complement $\mathbb{S}$ involves $(\mathbb{F}+\mathbb{C})^{-1}$ and is impractical to work with. For our stochastic unsteady problem, we consider mean-based preconditioners that use approximations to the Schur complement matrix

$$\mathbb{S}_0 = \mathbb{B}(\mathbb{F}_0 + \mathbb{C})^{-1}\mathbb{B}^T, \tag{5.38}$$

where the "mean" matrix $\mathbb{F}_0$ is block-diagonal with $\mathbb{F}_0^k$ as the kth diagonal block, and

$$\mathbb{F}_0^k = \tau^{-1}(I_{n_\xi} \otimes \boldsymbol{M}) + I_{n_\xi} \otimes \boldsymbol{A}_0 + I_{n_\xi} \otimes \boldsymbol{N}(\bar{w}_{h,1}^k). \tag{5.39}$$

This corresponds to taking only the first term in the two summations on the right-hand side of eq. (5.9). Since the gPC basis functions are orthonormal with $\langle\psi_r\psi_s\rangle = \delta_{rs}$ and $\psi_1 \equiv 1$, it follows $\langle\psi_s\rangle = \delta_{1s}$, and $G_0 = H_1 = I_{n_\xi}$. The matrices $\boldsymbol{A}_0$ and

$\boldsymbol{N}(\vec{w}_{h,1}^k)$ are constructed from the mean of ν and $\vec{w}_h^k$,

$$\langle \nu \rangle = \nu_0, \ \langle \vec{w}_h^k \rangle = \sum\nolimits_j w_{j1}^k \vec{\phi}_j(x) = \vec{w}_{h,1}^k. \tag{5.40}$$

The matrix $\mathbb{F}_0^k$ can be expressed as $I_{n_\xi} \otimes (\tau^{-1}\boldsymbol{M} + \boldsymbol{A}_0 + \boldsymbol{N}(\vec{w}_{h,1}^k))$ and this enables use of approximations associated with a deterministic problem. Now, similarly define $\mathbb{F}_{p,0}$ on the pressure space, with

$$\mathbb{F}_{p,0}^k = \tau^{-1}(I_{n_\xi} \otimes M_p) + I_{n_\xi} \otimes A_{p,0} + I_{n_\xi} \otimes N_p(\vec{w}_{h,1}^k). \tag{5.41}$$

Let $\mathbb{M} = I_{n_t} \otimes I_{n_\xi} \otimes \boldsymbol{M}$ and $\mathbb{M}_p = I_{n_t} \otimes I_{n_\xi} \otimes M_p$. Assuming the validity of eq. (5.33) it is easy to check that

$$\mathbb{M}_p^{-1}\mathbb{B}\mathbb{M}^{-1}\mathbb{F}_0 - \mathbb{M}_p^{-1}\mathbb{F}_{p,0}\mathbb{M}_p^{-1}\mathbb{B} \approx 0. \tag{5.42}$$

On the other hand, let $\mathbb{C}_p = -\tau^{-1}C_{n_t} \otimes I_{n_\xi} \otimes M_p$, so that $\mathbb{C}$ satisfies

$$\mathbb{M}_p^{-1}\mathbb{B}\mathbb{M}^{-1}\mathbb{C} - \mathbb{M}_p^{-1}\mathbb{C}_p\mathbb{M}_p^{-1}\mathbb{B} = 0. \tag{5.43}$$

Combining eq. (5.42) and eq. (5.43) gives an approximation to $\mathbb{S}_0$,

$$\mathbb{S}_0 = \mathbb{B}(\mathbb{F}_0 + \mathbb{C})^{-1}\mathbb{B}^T \approx \mathbb{M}_p(\mathbb{F}_{p,0} + \mathbb{C}_p)^{-1}\mathbb{B}\mathbb{M}^{-1}\mathbb{B}^T. \tag{5.44}$$

Then the mean-based PCD preconditioner is given as

$$\mathbb{S}_{\mathrm{PCD},0}^{-1} = (\mathbb{B}\mathbb{M}_*^{-1}\mathbb{B}^T)^{-1}(\mathbb{F}_{p,0} + \mathbb{C}_p)\mathbb{M}_{p*}^{-1}, \tag{5.45}$$

where $\mathbb{M}_* = I_{n_t} \otimes I_{n_\xi} \otimes \boldsymbol{M}_*$ and $\mathbb{M}_{p*} = I_{n_t} \otimes I_{n_\xi} \otimes M_{p*}$. Similarly from eq. (5.36), it holds that

$$\mathbb{F}_{p,0} + \mathbb{C}_p \approx (\mathbb{B}\mathbb{M}^{-1}(\mathbb{F}_0 + \mathbb{C})\mathbb{M}^{-1}\mathbb{B}^T)(\mathbb{B}\mathbb{M}^{-1}\mathbb{B})^{-1}\mathbb{M}_p. \tag{5.46}$$

Substituting $\mathbb{F}_{p,0} + \mathbb{C}_p$ in eq. (5.45) and replacement of the mass matrices with their diagonals gives the mean-based LSC preconditioner

$$\mathbb{S}_{\text{LSC},0}^{-1} = (\mathbb{B}\mathbb{M}_*^{-1}\mathbb{B}^T)^{-1}(\mathbb{B}\mathbb{M}_*^{-1}(\mathbb{F}_0 + \mathbb{C})\mathbb{M}_*^{-1}\mathbb{B}^T)(\mathbb{B}\mathbb{M}_*^{-1}\mathbb{B}^T)^{-1}. \tag{5.47}$$

The two mean-based preconditioners in eqs. (5.45) and (5.47) have the same form as for the deterministic problem, except that there is an extra term $\mathbb{C}$ or $\mathbb{C}_p$ from the all-at-once formulation. Computations associated with the two approximations to the Schur complement are also inexpensive. For example, $(\mathbb{B}\mathbb{M}_*^{-1}\mathbb{B}^T)^{-1} = I_{n_t} \otimes I_{n_\xi} \otimes (B\boldsymbol{M}_*B^T)^{-1}$, and this only requires solving a system with $B\boldsymbol{M}_*B^T$ a discrete Laplacian. Multiplications with the mean matrix $\mathbb{F}_0 + \mathbb{C}$ are reduced to its components (see eq. (5.22)),

$$\mathbb{F}_0 + \mathbb{C} = \tau^{-1}(I_{n_t} \otimes I_{n_\xi} \otimes \boldsymbol{M}) + I_{n_t} \otimes I_{n_\xi} \otimes \boldsymbol{A}_0 + \mathbb{N}_0 - \tau^{-1}(C_{n_t} \otimes I_{n_\xi} \otimes \boldsymbol{M}). \tag{5.48}$$

The matrix $\mathbb{N}_0$ is block-diagonal with $\mathbb{N}_0^k = I_{n_\xi} \otimes \boldsymbol{N}(\vec{w}_{h,1}^k)$ and can be expressed as a sum of Kronecker products of matrices as discussed in section 5.4.3,

$$\mathbb{N}_0(\boldsymbol{w}) = \sum_{\alpha_1,\alpha_2} \text{diag}(w_{\alpha_1}^{(1)}) \otimes (\underline{w}^{(2)}(\alpha_1, 1, \alpha_2) I_{n_\xi}) \otimes \boldsymbol{N}(\vec{w}_{\alpha_2}^{(3)}). \tag{5.49}$$

5.5.3 System solve with $\mathbb{F} + \mathbb{C}$

The application of the preconditioner $\mathscr{P}^{-1}$ in eq. (5.30) also involves solving a linear system asspciated with the (1,1) block $\mathbb{F} + \mathbb{C}$. For approximation, we also replace it with the mean matrix $\mathbb{F}_0 + \mathbb{C}$. This is a block-triangular matrix in the

form

$$\begin{pmatrix} \mathbb{F}_0^1 & & & & \\ -\tau^{-1}(I\otimes \boldsymbol{M}) & \mathbb{F}_0^2 & & & \\ & \ddots & \ddots & & \\ & & -\tau^{-1}(I\otimes \boldsymbol{M}) & \mathbb{F}_0^{n_t-1} & \\ & & & -\tau^{-1}(I\otimes \boldsymbol{M}) & \mathbb{F}_0^{n_t} \end{pmatrix}. \tag{5.50}$$

For such a system it is easy to compute matrix vector products and we again use a low-rank GMRES method for solving the system. This inner GMRES solver is preconditioned with $I_{n_t} \otimes I_{n_\xi} \otimes (\tau^{-1}\boldsymbol{M} + \boldsymbol{A}_0 + \boldsymbol{N}(\vec{w}_{h,1}^{\text{avg}}))$, where $\vec{w}_{h,1}^{\text{avg}}$ means the average of $\vec{w}_{h,1}^{k}$ over all time steps. The linear system has the form

$$(\mathbb{F}_0 + \mathbb{C})\boldsymbol{v} = \boldsymbol{y}, \tag{5.51}$$

and we note that this system need not to be solved accurately. In particular, with a stopping criterion $\|\boldsymbol{y} - (\mathbb{F}_0 + \mathbb{C})\boldsymbol{v}\|_2 \leq tol\|\boldsymbol{y}\|_2$, a relatively large stopping tolerance, e.g., $tol = 10^{-1}$, will suffice for the mean-based preconditioner $\mathscr{P}$ to be effective.

5.6 Numerical experiments

5.6.1 Benchmark problem

Consider a flow around a symmetric step where the spatial domain $\mathcal{D}$ is a two-dimensional rectangular duct with a symmetric expansion (see section 5.6.1). The Dirichlet inflow boundary conditions at $(-1, x_2)$, $|x_2| \leq 0.5$ are deterministic

and time-dependent, growing from zero to a steady parabolic profile,

$$\vec{u}_{\mathrm{D}}((-1,x_2),t) = \begin{pmatrix} 1-4x_2^2 \\ \\ 0 \end{pmatrix} (1-e^{-10t}). \tag{5.52}$$

Neumann boundary conditions $\nu\partial u_{x_1}/\partial x_1 = p$, $\partial u_{x_2}/\partial x_1 = 0$ are imposed at the outflow boundary $(12, x_2)$, $|x_2| \leq 1$, and no-flow conditions $\vec{u} = \vec{0}$ at the fixed walls $(x_1, \pm 1)$, $0 \leq x_1 \leq 12$; $(x_1, \pm 0.5)$, $-1 \leq x_1 \leq 0$; $(0, x_2)$, $0.5 \leq |x_2| \leq 1$. The initial conditions are zero everywhere for both $\vec{u}$ and p. The Taylor–Hood spatial discretization with biquadratic basis functions for the velocity space and bilinear basis functions for the pressure space is defined on a uniform grid of square elements with mesh size h, and it is constructed using the IFISS software package [82].

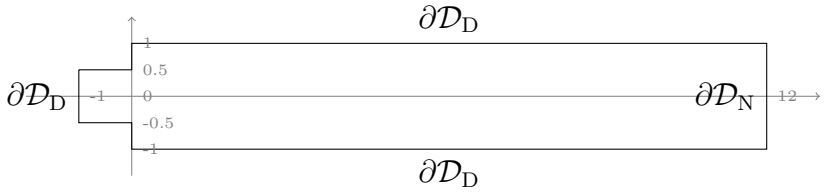


Figure 5.1: Symmetric step domain with boundary conditions.

The stochastic viscosity $\nu(x,\xi)$ is represented as a truncated KL expansion

$$\nu(x,\xi) = \nu_0\Big(1.0 + \sigma \sum_{l=1}^{m} \sqrt{\beta_l}\, a_l(x)\xi_l\Big). \tag{5.53}$$

The constants ν_0 and σ represent the mean and the standard deviation of the stochastic field. We use an exponential covariance function $c(x,y) = \exp(-\|x - y\|_1/b)$, where b is the correlation length. The pair $(\beta_l, a_l(x))$ is the lth largest eigenvalue and the corresponding eigenfunction of $c(x,y)$, satisfying

$$\int_{\mathcal{D}} c(x,y)a_l(y)\mathrm{d}y = \beta_l a_l(x). \tag{5.54}$$

This can be computed with a standard finite element method. The random variables $\{\xi_l\}_{l=1}^m$ are assumed to be independent and each of them uniformly distributed on the interval $[-\sqrt{3}, \sqrt{3}]$, so they have zero means and unit variances. For the stochastic Galerkin method, the basis functions $\{\psi_r\}_{r=1}^{n_\xi}$ are m-dimensional Legendre polynomials, with total degrees bounded by d_ψ. Then the number of stochastic basis functions is $n_\xi = (m + d_\psi)!/(m!d_\psi!)$. In the numerical experiments, unless otherwise stated, the parameter values associated with the discrete problem are chosen as in section 5.6.1. This gives a problem with dimensions $n_t = 64$, $n_\xi = 20$, $n_u = 2992$, $n_p = 461$, and $n_t n_\xi (n_u + n_p) = 4419840$. All computations are done in MATLAB 9.4.0 (R2018a) on a desktop with 64 GB memory.

Table 5.1: Parameter values for numerical experiments.

ν_0	σ	b	m	d_ψ	t_f	τ	h
1/50	0.01	4.0	3	3	1.0	2^{-6}	2^{-2}

5.6.2 Inexact Picard method

The main computational cost associated with Picard's method is to solve an all-at-once system eq. (5.17) at each step. In section 5.4 we discussed how to construct low-rank approximate solutions in tensor train format with much cheaper computations. To further reduce the cost, we adopt the idea of inexact Picard method [11], where the linear systems are solved inexactly to save unnecessary computational work. Let eq. (5.17) be denoted as $\mathscr{L}\boldsymbol{z}^{(i)} = \boldsymbol{r}^{(i-1)}$, and define the residual norm $\|\boldsymbol{r}_k\|_2 = \|\boldsymbol{r}^{(i-1)} - \mathscr{L}\boldsymbol{z}_k^{(i)}\|_2$ for an approximate solution $\boldsymbol{z}_k^{(i)}$. It was shown in [11]

that if the stopping criterion for the linear solve is given as

$$\|\boldsymbol{r}_k\|_2 \leq tol_{\text{gmres}}\|\boldsymbol{r}^{(i-1)}\|_2, \tag{5.55}$$

then Picard's method converges as long as $tol_{\text{gmres}} < 1$. This is especially helpful for our low-rank GMRES method. The best accuracy that the low-rank GMRES method can achieve is related to the truncation tolerance ϵ_{gmres} used in the algorithm (see fig. 5.2a). A relaxed stopping tolerance not only reduces the number of GMRES iterations, but it also allows use of larger truncation tolerances for tensor rank compressions, resulting in smaller ranks for the iterates and more efficient computations in the iterative solver. In the numerical tests, we set $tol_{\text{gmres}} = 10^{-1}$ and $\epsilon_{\text{gmres}} = 10^{-3}$. The same tolerances are used in the preconditioners for solving the linear system eq. (5.51). For the initial $\boldsymbol{u}^{(0)}$, $\boldsymbol{p}^{(0)}$, the Stokes problem is solved to satisfy $\|\boldsymbol{r}_k\|_2 \leq tol_{\text{gmres}}\|\boldsymbol{f}\|_2$ where $\boldsymbol{f}$ is the right-hand side of eq. (5.16).

When solving the linear system eq. (5.17), we also use an approximation to the convection matrix $\mathbb{N}$ to improve efficiency. As discussed in section 5.4.3, the matrix-vector product with the convection matrix $\mathbb{N}$ results in a dramatic rank increase from κ to κ^2. Unless κ is very small, a tensor train with rank κ^2 will require too much memory and also be expensive to work with. To overcome this difficulty, in the all-at-once system we use a low-rank approximation of $\boldsymbol{u}^{(i)}$ to construct $\mathbb{N}(\boldsymbol{u}^{(i)})$. Specifically, let

$$\tilde{\boldsymbol{u}}^{(i)} = \mathcal{T}_{\epsilon_{\text{conv}}}(\boldsymbol{u}^{(i)}) \tag{5.56}$$

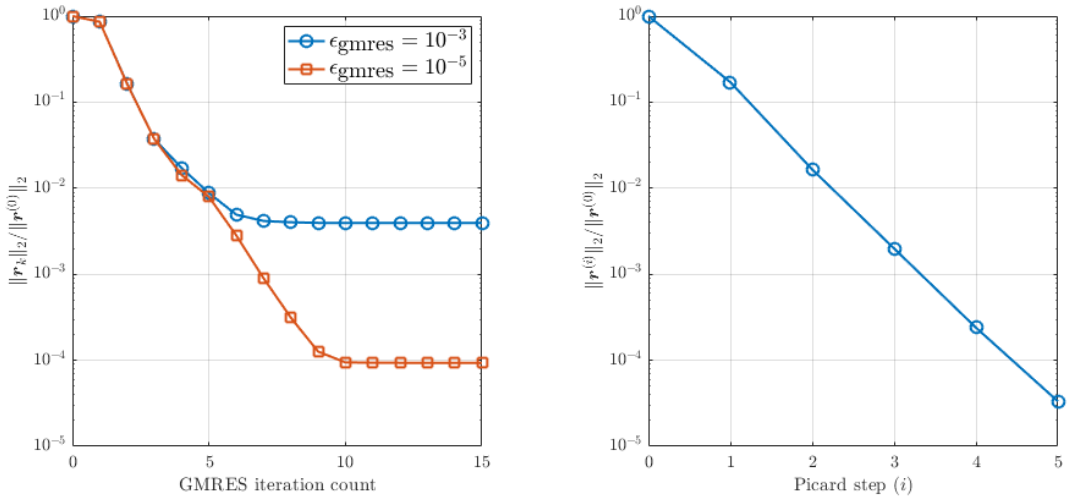
with some truncation tolerance ϵ_{conv}. Since $\tilde{\boldsymbol{u}}^{(i)}$ has smaller ranks than $\boldsymbol{u}^{(i)}$, the approximate convection matrix $\mathbb{N}(\tilde{\boldsymbol{u}}^{(i)})$ contains a smaller number of terms in eq. (5.28),

and thus the rank increase becomes less significant when computing matrix-vector product with it.

The choice of stopping and truncation tolerances are summarized in section 5.6.2. The stopping tolerance of Picard's method is $tol_{\text{picard}} = 10^{-4}$, and a small truncation tolerance $\epsilon_{\text{soln}} = 10^{-7}$ is used to compress the ranks of the approximate solutions $\boldsymbol{u}^{(i)}$ and $\boldsymbol{p}^{(i)}$ at each Picard step. It is shown in fig. 5.2b that, like the exact method, the inexact Picard method still exhibits a linear convergence rate. It takes 5 Picard steps to reach the required accuracy. Section 5.6.2 shows the ranks of the iterates at each Picard step. As the Picard iteration converges, the right-hand side of eq. (5.55) becomes smaller, and the corrections $\delta\boldsymbol{u}^{(i)}$ and $\delta\boldsymbol{p}^{(i)}$ computed from the low-rank GMRES method have increasing ranks. On the other hand, for the approximate solutions $\boldsymbol{u}^{(i)}$ and $\boldsymbol{p}^{(i)}$, their ranks drop to smaller values in the end. Also shown in fig. 5.3b are the ranks of $\tilde{\boldsymbol{u}}^{(i)}$ for constructing the approximate convection matrices. They have much smaller values than the ranks of $\boldsymbol{u}^{(i)}$.

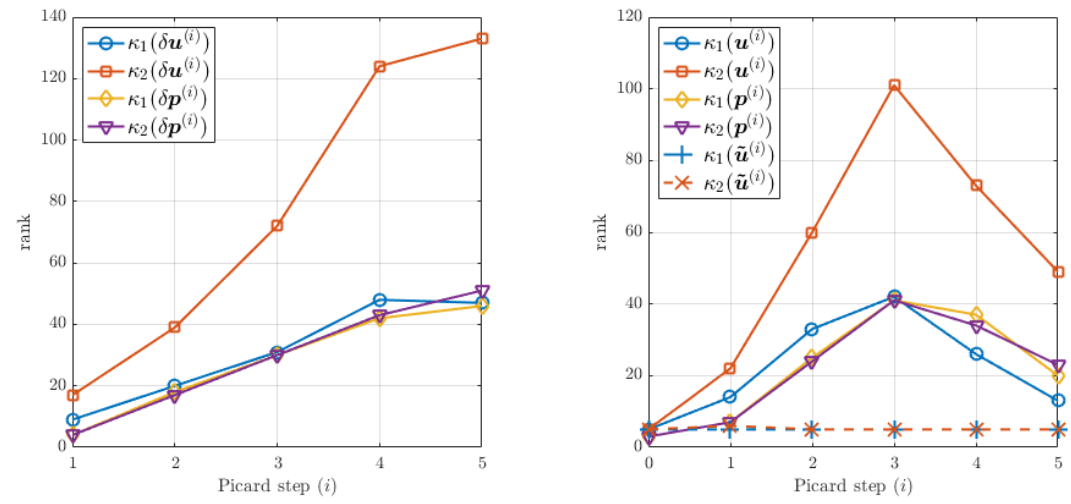
Table 5.2: Stopping and truncation tolerances.

GMRES stopping tolerance	$tol_{\text{gmres}} = 10^{-1}$
GMRES truncation tolerance	$\epsilon_{\text{gmres}} = 10^{-3}$
Picard stopping tolerance	$tol_{\text{picard}} = 10^{-4}$
Truncation tolerance for solutions	$\epsilon_{\text{soln}} = 10^{-7}$
Truncation tolerance for convection matrix	$\epsilon_{\text{conv}} = 10^{-3}$



(a) (b)

Figure 5.2: (a) Convergence of the low-rank GMRES method (at the first Picard step) with different truncation tolerances. (b) Convergence of the inexact Picard method.



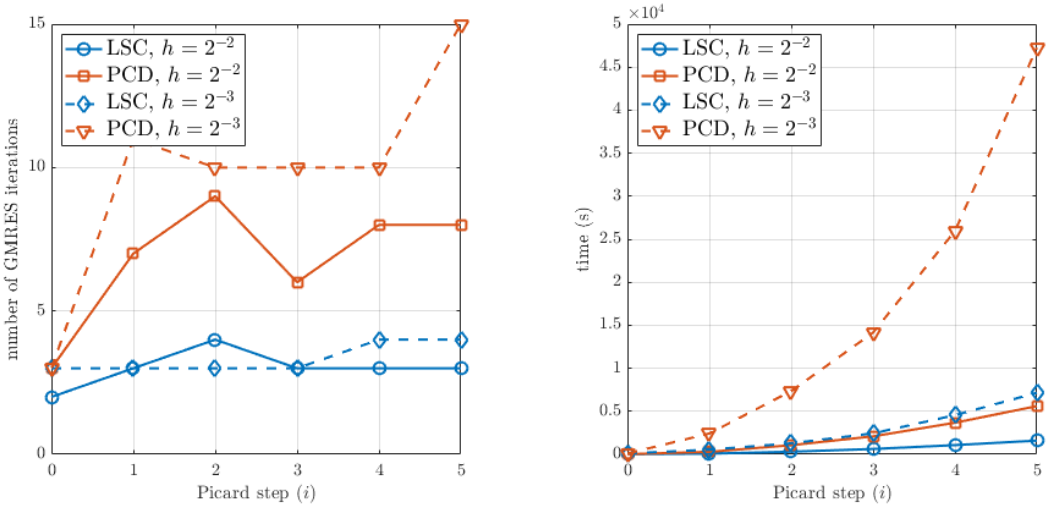
(a) (b)

Figure 5.3: (a) Ranks of corrections $\delta \boldsymbol{u}^{(i)}$ and $\delta \boldsymbol{p}^{(i)}$. (b) Ranks of approximate solutions $\boldsymbol{u}^{(i)}$ and $\boldsymbol{p}^{(i)}$, and ranks of $\tilde{\boldsymbol{u}}^{(i)}$ for convection matrix.

5.6.3 Numerical results

In the following, we examine the performance of the proposed algorithm in different settings. First of all, we compare the two mean-based preconditioners discussed in section 5.5. Section 5.6.3 shows the number of GMRES iterations required at each Picard step, and the associated computational costs when the two preconditioners are used. For two different mesh sizes, the PCD preconditioner results in larger numbers of GMRES iterations, and thus higher computational times, than the LSC preconditioner. It should also be noted that for both preconditioners, only a small number of GMRES iterations is needed for solving the linear system at each Picard step. This is partially due to the large stopping tolerance used in eq. (5.55). It shows that with inexact Picard method and effective preconditioners, the work required for GMRES is greatly reduced. The LSC preconditioner will be used for the numerical tests below.

Next, we test the algorithm with several variants of the benchmark problem determined by various values of parameters associated with it. Figure 5.5a shows the solution ranks and computational times for three different values of the standard deviation σ. When σ is smaller, the discrete solution can be approximated by a tensor train with smaller ranks, and it is also less expensive to solve the nonlinear problem. On the other hand, even for $\sigma = 0.1$, the low-rank solution takes much less storage than a full tensor. For example, the ranks of the approximate solution $\boldsymbol{u}^{(i)}$ are $\kappa_1 = 13$, $\kappa_2 = 93$. The ratio of storage requirements between such a tensor



(a) (b)

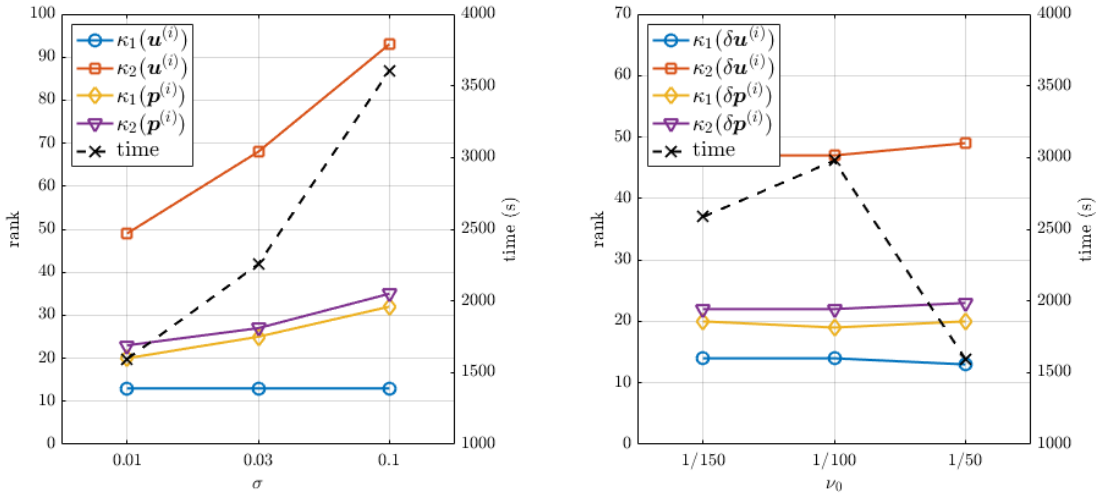
Figure 5.4: (a) Number of GMRES iterations at each Picard step. (b) Accumulative computational time after each Picard step.

train and a full tensor is

$$\frac{n_t\kappa_1 + n_\xi\kappa_1\kappa_2 + n_u\kappa_2}{n_t n_\xi n_u} = \frac{303268}{3829760} \approx 7.9\%. \tag{5.57}$$

The same quantities are plotted in fig. 5.5b for different values of the mean viscosity ν_0. The ranks and computational times are not significantly affected by ν_0.

Finally, the algorithm is applied to solve discrete problems with various mesh sizes h or time step sizes τ. It can be seen from fig. 5.6a that there is only a slight increase in the solution ranks as the spatial mesh is refined. It is also shown in fig. 5.6a that the computational time increases as $O(h^{-2})$. In other words, as the spatial mesh is refined, no extra computational burden is introduced except for the increased problem size. However, this is not the case for the time step size τ. The computational time increases much faster than $O(\tau^{-1})$ (see fig. 5.6b). This is due to solving the linear system eq. (5.51) within the preconditioner $\mathscr{P}$. As τ gets smaller, the off-diagonal blocks have larger values, and the block-diagonal preconditioner



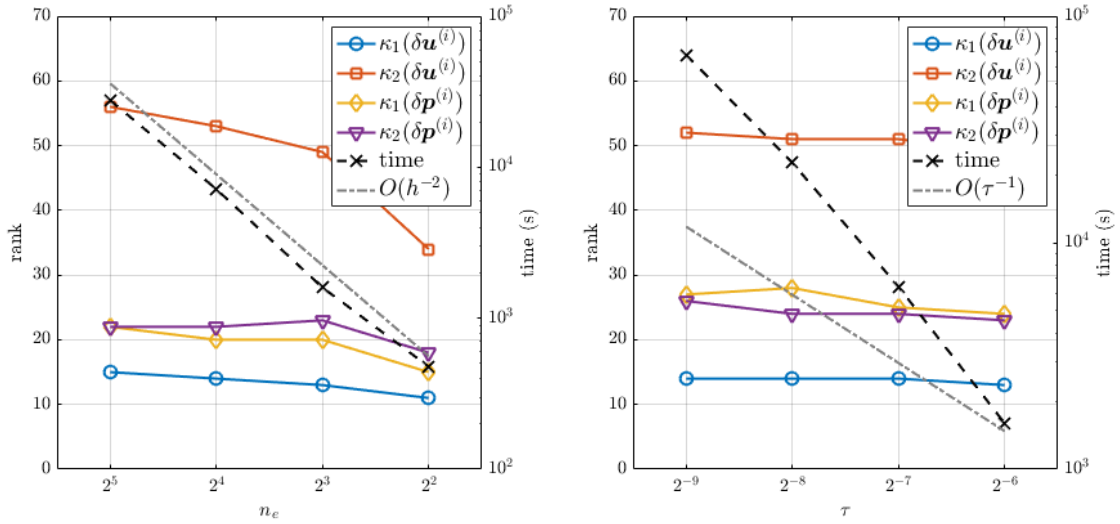
(a) (b)

Figure 5.5: Solution ranks and computational times for different values of σ and ν_0.

$I_{n_t} \otimes I_{n_\xi} \otimes (\tau^{-1}\boldsymbol{M} + \boldsymbol{A}_0 + \boldsymbol{N}(\vec{w}_{h,1}^{\text{avg}}))$ seems to be less effective. More inner iterations for solving this system results in higher computational costs.

5.7 Conclusions

In this chapter, we developed and studied efficient low-rank iterative methods for solving the time-dependent Navier–Stokes equations with a random viscosity. We considered an all-at-once formulation where the discrete solutions at all the time steps are solved together in a single system. To address the high storage and computational costs of this strategy, we used low-rank tensor approximations in a Newton–Krylov type algorithm. For the all-at-once system, we proposed two mean-based preconditioners using results from the deterministic problem. The computational costs were further reduced with inexact Picard method and approximate convection matrices. It was shown in the numerical experiments that the low-rank



(a) (b)

Figure 5.6: Solution ranks and computational times for different values of h and τ. In (a), $n_e = 2/h$ is the number of elements in the vertical interval $[-1, 1]$ of the domain $\mathcal{D}$.

method is able to solve the nonlinear problem efficiently and the discrete solutions have small tensor ranks.

Chapter 6: Concluding Remarks

In this thesis we developed efficient computational methods for stochastic PDEs, including the linear diffusion problem and nonlinear eigenvalue problems and time-dependent Navier–Stokes equations. The stochastic Galerkin method generates surrogate solutions to the stochastic models, which greatly facilitate the task of uncertainty quantification. On the other hand, to compute the surrogates in general entails solving large-size and computationally challenging algebraic systems. We designed iterative algorithms to construct low-rank approximations to the solutions, and demonstrated that with low-rank methods the systems can be solved with much lower costs than the problem size suggests.

For the stochastic diffusion problem, we developed a low-rank multigrid solver for the linear system obtained from stochastic Galerkin discretization. With a linear parametrization of the input data via KL expansion, the resulting system can be expressed as a sum of Kronecker products of smaller matrices. Such a structure enables cheap computations when the quantities in an iterative algorithm is represented as low-rank matrices. By combing low-rank matrix approximations with the multigrid method, and repeatedly applying a truncation operation to compress the ranks of intermediate quantities, the proposed method was shown to result in

significant cost savings. For a better understanding about the effect of truncation, we conducted a rigorous error analysis for the method. We remark that the low-rank solver handles problems with small variances very well, where the discrete solutions can be well approximated by low-rank matrices. Also, a more efficient truncation procedure will be helpful to further improv efficiency.

The stochastic inverse subspace iteration method for computing smallest eigenvalues and corresponding eigenvectors requires solving a large linear system at each step of iteration. We developed a low-rank variant of the algorithm where the linear systems are efficiently solved with a low-rank multigrid or Krylov subspace method. With low-rank approximation, the computational costs associated with the Gram–Schmidt process and the Rayleigh quotient construction are also greatly reduced. For a diffusion problem with poorly separated eigenvalues, we showed that a Rayleigh–Ritz procedure can be used to significantly improve the accuracy of the solution. In a Stokes model, we applied the low-rank method for an eigenvalue problem derived from the inf-sup constant, and a saddle-point system was solved at each iteration. It was demonstrated in the numerical experiments that the proposed method produces accurate results and uses less memory and computational work than its full-rank counterpart.

For the Navier–Stokes problem we designed a Newton–Krylov type solver with low-rank tensor approximations. With an all-at-once treatment of time integration and stochastic Galerkin method for the stochastic part, the discrete solution is represented as a three-dimensional tensor. Again, the Kronecker product structure of the system and the low-rank tensor train format allow cheap computations for a

large-size problem. Low-rank iterative methods typically encounter rank growth for intermediate quantities, and for a nonlinear problem this may be more dramatic. To address this difficulty, we adopted the idea of inexact Picard method where the linear system at each nonlinear step is only solved inexactly to save unnecessary computational work. Also, we derived mean-based preconditioners for the system so the Krylov subspace method has faster convergence and requires a small number of iterations. These techniques make the low-rank iterative solver efficient in handling large-size problems. As a side note, it will be interesting to see how one can borrow the advantages of the optimization-based DMRG-type methods to compute low-rank approximations for the Navier–Stokes equations.

Bibliography

[1] R. Andreev and C. Schwab, *Sparse tensor approximation of parametric eigenvalue problems*, in Numerical Analysis of Multiscale Problems, I. G. Graham, T. Y. Hou, O. Lakkis, and R. Scheichl, eds., Springer, Berlin, 2012, pp. 203–241.

[2] R. Andreev and C. Tobler, *Multilevel preconditioning and low-rank tensor iteration for space-time simultaneous discretizations of parabolic PDEs*, Numerical Linear Algebra with Applications, 22 (2015), pp. 317–337.

[3] I. Babuška, R. Tempone, and G. E. Zouraris, *Galerkin finite element approximations of stochastic elliptic partial differential equations*, SIAM Journal on Numerical Analysis, 42 (2004), pp. 800–825.

[4] J. Ballani and L. Grasedyck, *A projection method to solve linear systems in tensor format*, Numerical Linear Algebra with Applications, 20 (2013), pp. 27–43.

[5] P. Benner, S. Dolgov, A. Onwunta, and M. Stoll, *Low-rank solvers for unsteady Stokes–Brinkman optimal control problem with random data*, Computer Methods in Applied Mechanics and Engineering, 304 (2016), pp. 26–54.

[6] ——, *Solving optimal control problems governed by random Navier–Stokes equations using low-rank methods*. http://arxiv.org/abs/1703.06097, Mar. 2017.

[7] P. Benner, A. Onwunta, and M. Stoll, *Low-rank solution of unsteady diffusion equations with stochastic coefficients*, SIAM/ASA Journal on Uncertainty Quantification, 3 (2015), pp. 622–649.

[8] ——, *Block-diagonal preconditioning for optimal control problems constrained by PDEs with uncertain inputs*, SIAM Journal on Matrix Analysis and Applications, 37 (2016), pp. 491–518.

[9] ——, *A low-rank inexact Newton–Krylov method for stochastic eigenvalue problems*, Computational Methods in Applied Mathematics, 19 (2019), pp. 5–22.

[10] P. Benner, Y. Qiu, and M. Stoll, *Low-rank eigenvector compression of posterior covariance matrices for linear Gaussian inverse problems*, SIAM/ASA Journal on Uncertainty Quantification, 6 (2018), pp. 965–989.

[11] P. Birken, *Termination criteria for inexact fixed-point schemes*, Numerical Linear Algebra with Applications, 22 (2015), pp. 702–716.

[12] Å. Björck and G. H. Golub, *Numerical methods for computing angles between linear subspaces*, Mathematics of Computation, 27 (1973), pp. 579–594.

[13] A. Brandt, S. McCormick, and J. Ruge, *Algebraic multigrid (AMG) for sparse matrix equations*, in Sparsity and its Applications, D. Evans, ed., Cambridge University Press, Cambridge, 1984, pp. 257–284.

[14] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A Multigrid Tutorial*, SIAM, Philadelphia, second ed., 2000.

[15] A. Cohen, R. DeVore, and C. Schwab, *Convergence rates of best N-term Galerkin approximations for a class of elliptic sPDEs*, Foundations of Computational Mathematics, 10 (2010), pp. 615–646.

[16] R. S. Dembo, S. C. Eisenstat, and T. Steihaug, *Inexact Newton methods*, SIAM Journal on Numerical Analysis, 19 (1982), pp. 400–408.

[17] J. Dick, F. Y. Kuo, and I. H. Sloan, *High-dimensional integration: the quasi-Monte Carlo way*, Acta Numerica, 22 (2013), pp. 133–288.

[18] S. V. Dolgov, *TT-GMRES: solution to a linear system in the structured tensor format*, Russian Journal of Numerical Analysis and Mathematical Modelling, 28 (2013), pp. 149–172.

[19] S. V. Dolgov and D. V. Savostyanov, *Alternating minimal energy methods for linear systems in higher dimensions*, SIAM Journal on Scientific Computing, 36 (2014), pp. A2248–A2271.

[20] C. Eckart and G. Young, *The approximation of one matrix by another of lower rank*, Psychometrika, 1 (1936), pp. 211–218.

[21] M. Eiermann, O. G. Ernst, and E. Ullmann, *Computational aspects of the stochastic finite element method*, Computing and Visualization in Science, 10 (2007), pp. 5–15.

[22] S. C. Eisenstat and H. F. Walker, *Choosing the forcing terms in an inexact Newton method*, SIAM Journal on Scientific Computing, 17 (1996), pp. 16–32.

[23] H. Elman and D. Furnival, *Solving the stochastic steady-state diffusion problem using multigrid*, IMA Journal of Numerical Analysis, 27 (2007), pp. 675–688.

[24] H. Elman, M. Mihajlović, and D. Silvester, *Fast iterative solvers for buoyancy driven flow problems*, Journal of Computational Physics, 230 (2011), pp. 3900–3914.

[25] H. C. Elman, O. G. Ernst, D. P. OLeary, and M. Stewart, *Efficient iterative algorithms for the stochastic finite element method with application to acoustic scattering*, Computer Methods in Applied Mechanics and Engineering, 194 (2005), pp. 1037–1055.

[26] H. C. Elman, D. J. Silvester, and A. J. Wathen, *Finite Elements and Fast Iterative Solvers: With Applications in Incompressible Fluid Dynamics*, Oxford University Press, Oxford, second ed., 2014.

[27] H. C. Elman and T. Su, *A low-rank multigrid method for the stochastic steady-state diffusion problem*, SIAM Journal on Matrix Analysis and Applications, 39 (2018), pp. 492–509.

[28] ——, *Low-rank solution methods for stochastic eigenvalue problems*. http://arxiv.org/abs/1803.03717, Mar. 2018.

[29] O. G. Ernst and E. Ullmann, *Stochastic Galerkin matrices*, SIAM Journal on Matrix Analysis and Applications, 31 (2010), pp. 1848–1872.

[30] I. Fumagalli, A. Manzoni, N. Parolini, and M. Verani, *Reduced basis approximation and a posteriori error estimates for parametrized elliptic eigenvalue problems*, ESAIM: Mathematical Modelling and Numerical Analysis, 50 (2016), pp. 1857–1885.

[31] M. J. Gander and M. Neumüller, *Analysis of a new space-time parallel multigrid algorithm for parabolic problems*, SIAM Journal on Scientific Computing, 38 (2016), pp. A2173–A2208.

[32] T. Gerstner and M. Griebel, *Numerical integration using sparse grids*, Numerical Algorithms, 18 (1998), pp. 209–232.

[33] R. Ghanem and D. Ghosh, *Efficient characterization of the random eigenvalue problem in a polynomial chaos decomposition*, International Journal for Numerical Methods in Engineering, 72 (2007), pp. 486–504.

[34] R. G. Ghanem and P. D. Spanos, *Stochastic Finite Elements: A Spectral Approach*, Springer-Verlag, New York, 1991.

[35] M. B. Giles, *Multilevel Monte Carlo methods*, Acta Numerica, 24 (2015), pp. 259–328.

[36] G. H. Golub and Q. Ye, *Inexact inverse iteration for generalized eigenvalue problems*, BIT Numerical Mathematics, 40 (2000), pp. 671–684.

[37] L. Grasedyck, *Hierarchical singular value decomposition of tensors*, SIAM Journal on Matrix Analysis and Applications, 31 (2010), pp. 2029–2054.

[38] L. Grasedyck, D. Kressner, and C. Tobler, *A literature survey of low-rank tensor approximation techniques*, GAMM-Mitteilungen, 36 (2013), pp. 53–78.

[39] P. M. Gresho, D. F. Griffiths, and D. J. Silvester, *Adaptive time-stepping for incompressible flow part I: Scalar advection-diffusion*, SIAM Journal on Scientific Computing, 30 (2008), pp. 2018–2054.

[40] W. Hackbusch, *Multi-Grid Methods and Applications*, Springer-Verlag, Berlin, 1985.

[41] ——, *Solution of linear systems in high spatial dimensions*, Computing and Visualization in Science, 17 (2015), pp. 111–118.

[42] W. Hackbusch, B. N. Khoromskij, and E. E. Tyrtyshnikov, *Approximate iterations for structured matrices*, Numerische Mathematik, 109 (2008), pp. 365–383.

[43] H. Hakula, V. Kaarnioja, and M. Laaksonen, *Approximate methods for stochastic eigenvalue problems*, Applied Mathematics and Computation, 267 (2015), pp. 664–681.

[44] H. Hakula and M. Laaksonen, *Asymptotic convergence of spectral inverse iterations for stochastic eigenvalue problems*. http://arxiv.org/abs/1706.03558, June 2017.

[45] M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, Journal of Research of the National Bureau of Standards, 49 (1952), pp. 409–436.

[46] S. Holtz, T. Rohwedder, and R. Schneider, *The alternating linear scheme for tensor optimization in the tensor train format*, SIAM Journal on Scientific Computing, 34 (2012), pp. A683–A713.

[47] T. Horger, B. Wohlmuth, and T. Dickopf, *Simultaneous reduced basis approximation of parameterized elliptic eigenvalue problems*, ESAIM: Mathematical Modelling and Numerical Analysis, 51 (2017), pp. 443–465.

[48] D. B. P. Huynh, G. Rozza, S. Sen, and A. T. Patera, *A successive constraint linear optimization method for lower bounds of parametric coercivity and inf-sup stability constants*, Comptes Rendus Mathematique, 345 (2007), pp. 473–478.

[49] D. A. Kay, P. M. Gresho, D. F. Griffiths, and D. J. Silvester, *Adaptive time-stepping for incompressible flow part II: Navier–Stokes equations*, SIAM Journal on Scientific Computing, 32 (2010), pp. 111–128.

[50] C. T. Kelly, *Iterative Methods for Linear and Nonlinear Equations*, SIAM, Philadelphia, 1995.

[51] B. N. Khoromskij and C. Schwa, *Tensor-structured Galerkin approximation of parametric and stochastic elliptic PDEs*, SIAM Journal on Scientific Computing, 33 (2011), pp. 364–385.

[52] A. Klimke and B. Wohlmuth, *Algorithm 847: SPINTERP: Piecewise multilinear hierarchical sparse grid interpolation in MATLAB*, ACM Transactions on Mathematical Software, 31 (2005), pp. 561–579.

[53] O. M. Knio and O. P. Le Maître, *Uncertainty propagation in CFD using polynomial chaos decomposition*, Fluid Dynamics Research, 38 (2006), pp. 616–640.

[54] A. V. Knyazev, *Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method*, SIAM Journal on Scientific Computing, 23 (2001), pp. 517–541.

[55] T. G. Kolda and B. W. Bader., *Tensor decompositions and applications*, SIAM Review, 51 (2009), pp. 455–500.

[56] D. Kressner and C. Tobler, *Low-rank tensor Krylov subspace methods for parametrized linear systems*, SIAM Journal of Matrix Analysis and Applications, 32 (2011), pp. 1288–1316.

[57] D. Kressner and C. Tobler, *Preconditioned low-rank methods for high-dimensional elliptic PDE eigenvalue problems*, Computational Methods in Applied Mathematics, 11 (2011), pp. 363–381.

[58] Y.-L. Lai, K.-Y. Lin, and W.-W. Lin, *An inexact inverse iteration for large sparse eigenvalue problems*, Numerical Linear Algebra with Applications, 4 (1997), pp. 425–437.

[59] O. P. Le Maître and O. M. Knio, *Spectral Methods for Uncertainty Quantification: With Applications to Computational Fluid Dynamics*, Springer, Dordrecht, 2010.

[60] K. Lee and H. C. Elman, *A preconditioned low-rank projection method with a rank-reduction scheme for stochastic partial differential equations*, SIAM Journal on Scientific Computing, 39 (2017), pp. 828–850.

[61] M. Loève, *Probability Theory*, Van Nostrand, New York, 1960.

[62] G. J. Lord, C. E. Powell, and T. Shardlow, *An Introduction to Computational Stochastic PDEs*, Cambridge University Press, New York, 2014.

[63] L. Machiels, Y. Maday, I. B. Oliveira, A. T. Patera, and D. V. Rovas, *Output bounds for reduced-basis approximations of symmetric positive definite eigenvalue problems*, Comptes Rendus de l'Académie des Sciences - Series I - Mathematics, 331 (2000), pp. 153–158.

[64] Y. Maday and E. M. Rønquist, *Parallelization in time through tensor-product space–time solvers*, Comptes Rendus Mathematique, 346 (2008), pp. 113–118.

[65] H. G. Matthies and E. Zander, *Solving stochastic systems with low-rank tensor compression*, Linear Algebra and its Applications, 436 (2012), pp. 3819–3838.

[66] E. McDonald, J. Pestana, and A. Wathen, *Preconditioning and iterative solution of all-at-once systems for evolutionary partial differential equations*, SIAM Journal on Scientific Computing, 40 (2018), pp. A1012–A1033.

[67] H. Meidani and R. Ghanem, *Spectral power iterations for the random eigenvalue problem*, AIAA Journal, 52 (2014), pp. 912–925.

[68] H. N. Najm, *Uncertainty quantification and polynomial chaos techniques in computational fluid dynamics*, Annual Review of Fluid Mechanics, 41 (2009), pp. 35–52.

[69] N. Ngoc Cuong, K. Veroy, and A. T. Patera, *Certified real-time solution of parametrized partial differential equations*, in Handbook of Materials Modeling, S. Yip, ed., Springer Netherlands, Dordrecht, 2005, pp. 1529–1564.

[70] I. V. Oseledets, *Tensor-train decomposition*, SIAM Journal on Scientific Computing, 33 (2011), pp. 2295–2317.

[71] I. V. Oseledets, S. Dolgov, V. Kazeev, D. Savostyanov, O. Lebedeva, P. Zhlobich, T. Mach, and L. Song, *TT-Toolbox, Ver. 2.2.* http://github.com/oseledets/TT-Toolbox.

[72] C. C. Paige and M. A. Saunders, *Solution of sparse indefinite systems of linear equations*, SIAM Journal on Numerical Analysis, 12 (1975), pp. 617–629.

[73] M. F. Pellissetti and R. G. Ghanem, *Iterative solution of systems of linear equations arising in the context of stochastic finite elements*, Advances in Engineering Software, 31 (2000), pp. 607–616.

[74] C. E. Powell and H. C. Elman, *Block-diagonal preconditioning for spectral stochastic finite-element systems*, IMA Journal of Numerical Analysis, 29 (2009), pp. 350–375.

[75] C. E. Powell and D. J. Silvester, *Preconditioning steady-state Navier–Stokes equations with random data*, SIAM Journal on Scientific Computing, 34 (2012), pp. A2482–A2506.

[76] H. J. Pradlwarter, G. I. Schuëller, and G. S. Szekely, *Random eigenvalue problems for large systems*, Computers & Structures, 80 (2002), pp. 2415–2424.

[77] J. W. Ruge and K. Stüben, *Algebraic multigrid*, in Multigrid Methods, S. F. McCormick, ed., vol. 3 of Frontiers in Applied Mathematics, SIAM, Philadelphia, 1987, pp. 73–130.

[78] Y. Saad, *Iterative Methods for Sparse Linear Systems*, SIAM, Philadelphia, second ed., 2003.

[79] ——, *Numerical Methods for Large Eigenvalue Problems*, SIAM, Philadelphia, second ed., 2011.

[80] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM Journal on Scientific and Statistical Computing, 7 (1986), pp. 856–869.

[81] U. Schollwöck, *The density-matrix renormalization group*, Reviews of Modern Physics, 77 (2005), pp. 259–315.

[82] D. Silvester, H. Elman, and A. Ramage, *Incompressible Flow and Iterative Solver Software (IFISS), Ver. 3.5.* http://www.manchester.ac.uk/ifiss, Sept. 2016.

[83] D. J. Silvester and V. Simoncini, *An optimal iterative solver for symmetric indefinite systems stemming from mixed approximation*, ACM Transactions on Mathematical Software, 37 (2011), p. 42.

[84] P. Sirković and D. Kressner, *Subspace acceleration for large-scale parameter-dependent Hermitian eigenproblems*, SIAM Journal on Matrix Analysis and Applications, 37 (2016), pp. 695–718.

[85] D. C. Sorensen, *Implicit application of polynomial filters in a k-step Arnoldi method*, SIAM Journal on Matrix Analysis and Applications, 13 (1992), pp. 357–385.

[86] B. Sousedík and H. C. Elman, *Inverse subspace iteration for spectral stochastic finite element methods*, SIAM/ASA Journal on Uncertainty Quantification, 4 (2016), pp. 163–189.

[87] ——, *Stochastic Galerkin methods for the steady-state Navier–Stokes equations*, Journal of Computational Physics, 316 (2016), pp. 435–452.

[88] G. W. Stewart, *Accelerating the orthogonal iteration for the eigenvectors of a Hermitian matrix*, Numerische Mathematik, 13 (1969), pp. 362–376.

[89] G. W. Stewart, *Matrix Algorithms: Volume II: Eigensystems*, SIAM, Philadelphia, 2001.

[90] M. Stoll and T. Breiten, *A low-rank in time approach to PDE-constrained optimization*, SIAM Journal on Scientific Computing, 37 (2015), pp. B1–B29.

[91] T. Sullivan, *Introduction to Uncertainty Quantification*, Springer, Cham, 2015.

[92] E. Ullmann, *A Kronecker product preconditioner for stochastic Galerkin finite element discretizations*, SIAM Journal on Scientific Computing, 32 (2010), pp. 923–946.

[93] C. V. Verhoosel, M. A. Gutiérrez, and S. J. Hulshoff, *Iterative solution of the random eigenvalue problem with application to spectral stochastic finite element systems*, International Journal for Numerical Methods in Engineering, 68 (2006), pp. 401–424.

[94] K. Veroy and A. T. Patera, *Certified real-time solution of the parametrized steady incompressible Navier–Stokes equations: rigorous reduced-basis a posteriori error bounds*, International Journal for Numerical Methods in Fluids, 47 (2005), pp. 773–788.

[95] D. Xiu, *Numerical Methods for Stochastic Computations: A Spectral Method Approach*, Princeton University Press, Princeton, 2010.

[96] D. Xiu and G. E. Karniadakis, *The Wiener–Askey polynomial chaos for stochastic differential equations*, SIAM Journal on Scientific Computing, 24 (2002), pp. 619–644.

[97] ——, *Modeling uncertainty in flow simulations via generalized polynomial chaos*, Journal of Computational Physics, 187 (2003), pp. 137–167.